サンヨーパーソナルコンピュータ　SANYO

MSX プログラミング入門

（MSX　BASIC活用書）

WAVYシリーズ

MSX, MSX2

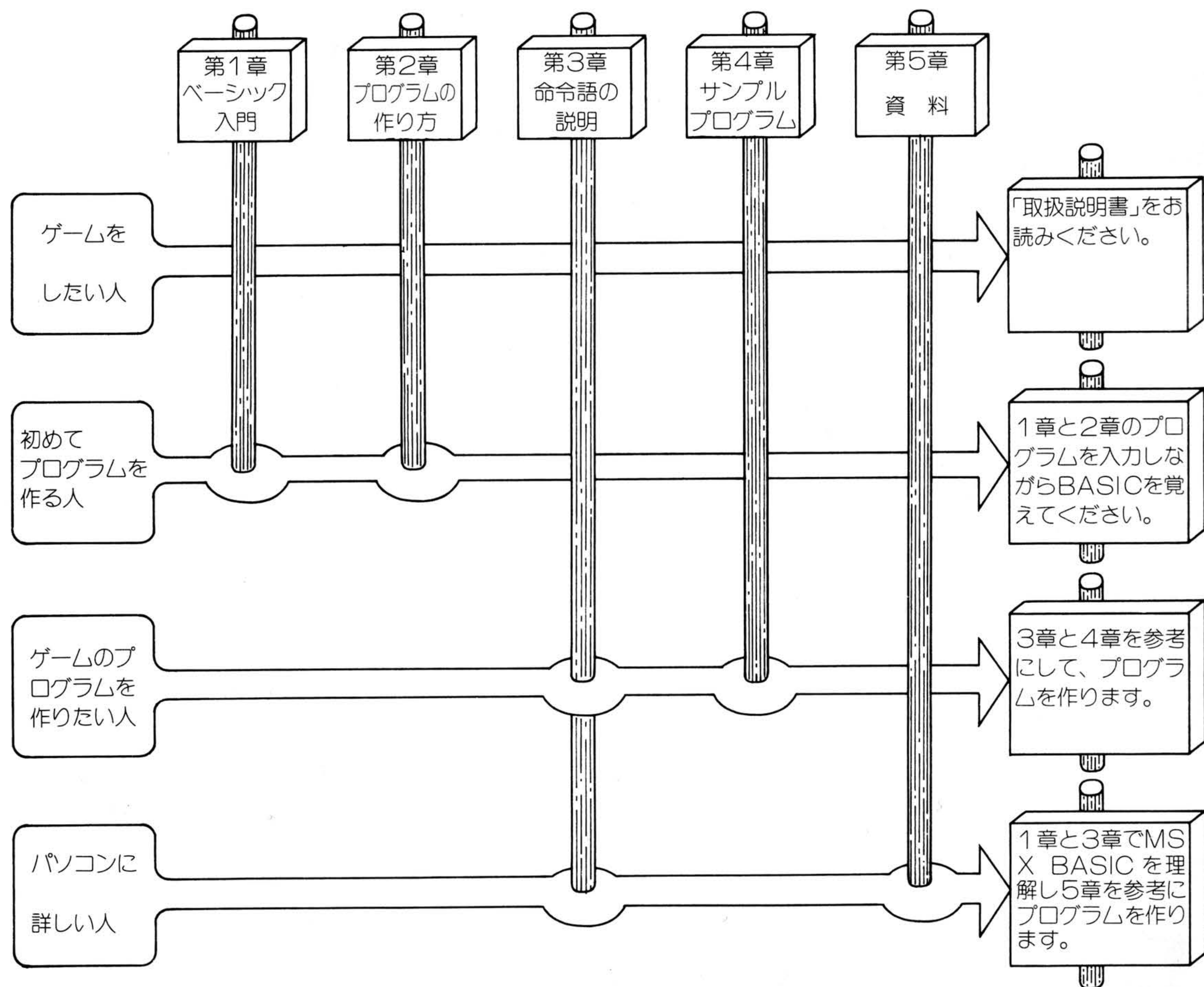
MSX プログラミング入門

MSX

MSX 2

ご注意

はじめに

このMSX（エムエスエックス）プログラミング入門は、MSXパーソナルコンピュータ（以後パソコンと略します）に使われているプログラム言語（MSX BASIC）の説明書です。この活用書は次の5つの章に分かれていますので、取扱説明書とともにパソコンの使用目的に合わせてお読みください。
別冊の「取扱説明書」では、パソコンや周辺機器の使いかたなどについてわかりやすく説明しています。

第1章 ベーシック入門
第2章 プログラムの作り方
第3章 命令語の説明
第4章 サンプルプログラム
第5章 資料

ゲームをしたい人 → 「取扱説明書」をお読みください。

初めてプログラムを作る人（第1章・第2章） → 1章と2章のプログラムを入力しながらBASICを覚えてください。

ゲームのプログラムを作りたい人（第3章・第4章） → 3章と4章を参考にして、プログラムを作ります。

パソコンに詳しい人（第3章・第5章） → 1章と3章でMSX BASICを理解し5章を参考にプログラムを作ります。

総　目　次

MSX2、MSX2+について

MSXパソコンには、MSXマークまたはMSX2、MSX2+マークのついているものがあります。
MSX2のパソコンは、MSXのパソコン機能を拡張したもので、MSXおよびMSX2マークのついたカートリッジなどのソフトウエアをそのまま使うことができます。
MSX2+のパソコンは、MSX2の機能をさらに拡張したものです。MSXおよびMSX2のマークのついたソフトウエアもそのまま使えます。
この「MSXプログラミング入門」では、MSX、MSX2の機能について説明しています。MSX2+のパソコンでMSX、MSX2の機能を使うときはこの本をご覧ください。
MSX2+用に拡張された命令については、MSX2+パソコンに付属の「拡張BASIC説明書」をご覧ください。

拡張機能	MSX2+の機能
拡張機能	MSX2の機能
MSXの機能	

第1章　目　　次

第2章　目　　次

第2　プログラムの作り方

〈簡単な絵を描き、音を出します〉

〈プログラムを楽しくするために〉

機能別索引

プログラムの作成・制御／状態設定

プログラムの作成

プログラムの制御

状態の設定

割り込み／分岐

割込み

画面への出力・制御／音を出す

画面の制御

テキスト画面への出力

グラフィック画面への出力

スプライト機能

音を出す

周辺装置との入出力

ファイルとしての入出力

キーボードからの入力

プリンタへの出力

カセットテープの入出力

ジョイスティックからの入力

タッチパネル・パドルからの入力

I／Oポートの入出力

機械語プログラム

プログラムの作成・実行

MSX Disk BASICのコマンド

プログラムの入出力

機能	コマンド	ページ
機械語プログラムをロードする	BLOAD	56
機械語プログラムをセーブする	BSAVE	57
プログラムの混合	MERGE	115
プログラムをアスキー形式でロードする	LOAD	110
プログラムをアスキー形式でセーブする	SAVE	153

ファイルの入出力

機能	コマンド	ページ
ファイルを開く	OPEN	126
ファイルを閉じる	CLOSE	64
ファイルの最後を調べる	EOF	85
ディスクの容量を調べる	DSKF	84
ファイルの位置を得る	LOC	110
ファイルの大きさを得る	LOF	112
バッファに1レコードを読み込む	GET	92
バッファの領域割り当て	FIELD	88
指定された長さの文字を得る	INPUT $	99
データを読み込む	INPUT #	98
1行分のデータを文字変数へ読み込む	LINE INPUT	97
バッファにデータを転送する	LSET/RSET	114
データを出力する	PRINT #/PRINT # USING	143、144
バッファのデータをファイルへ出力する	PUT	145
文字列を数値データに変換する	CVI/CVS/CVD	76
数値を内部表現のキャラクタコードに変換する	MKI $/MKS $/MKD $	117

ファイルの制御

機能	コマンド	ページ
ディスクをコピーする	COPY	73
ファイル名の表示	FILES	89
ディスケットの初期化	FORMAT	91
ファイルを消去する	KILL	104
ファイル名を変更する	NAME	118
ディスクからファイルをロードし、実行する	RUN	152

MSX2のコマンド

RAMディスク機能

機能	コマンド	ページ
使用を宣言し、初期化する	CALL MEMINI	58
ファイル名の表示	CALL MFILES	58
ファイルの消去	CALL MKILL	58
ファイル名の変更	CALL MNAME	58

時計機能

機能	コマンド	ページ
日付を得る	GET DATE	93
時刻を得る	GET TIME	93
日付を設定する	SET DATE	157
時刻を設定する	SET TIME	158

グラフィック機能

機能	コマンド	ページ
スプライトの色設定	COLOR SPRITE	69
カラーコードの設定	COLOR	65
画面データの転送	COPY	71
ディジタイズ(スチル)の実行	COPY SCREEN	74
漢字の表示	PUT KANJI	145
画面の設定	SCREEN	154
画面状態の設定	SET	156

第1章
ベーシック入門

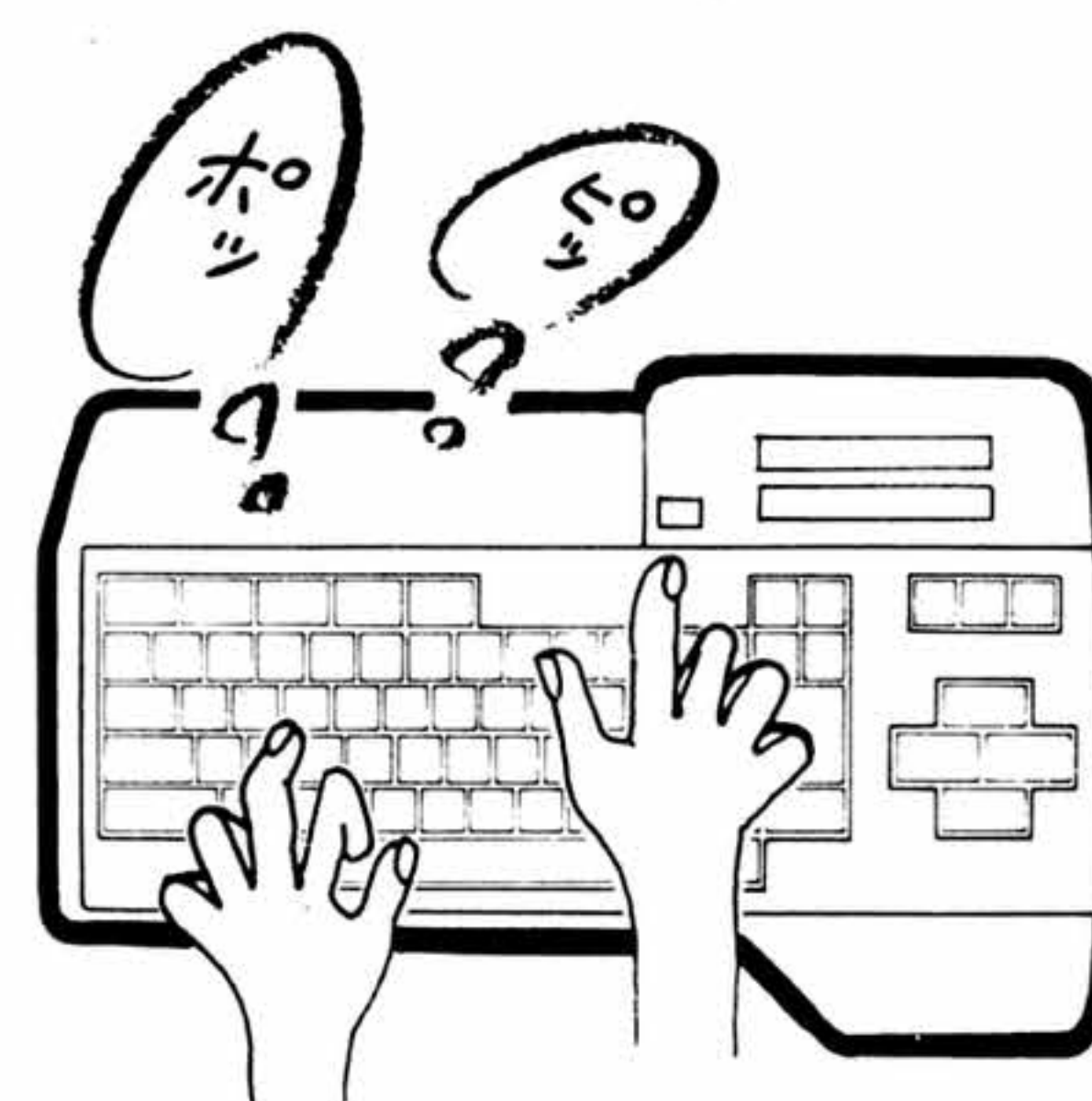
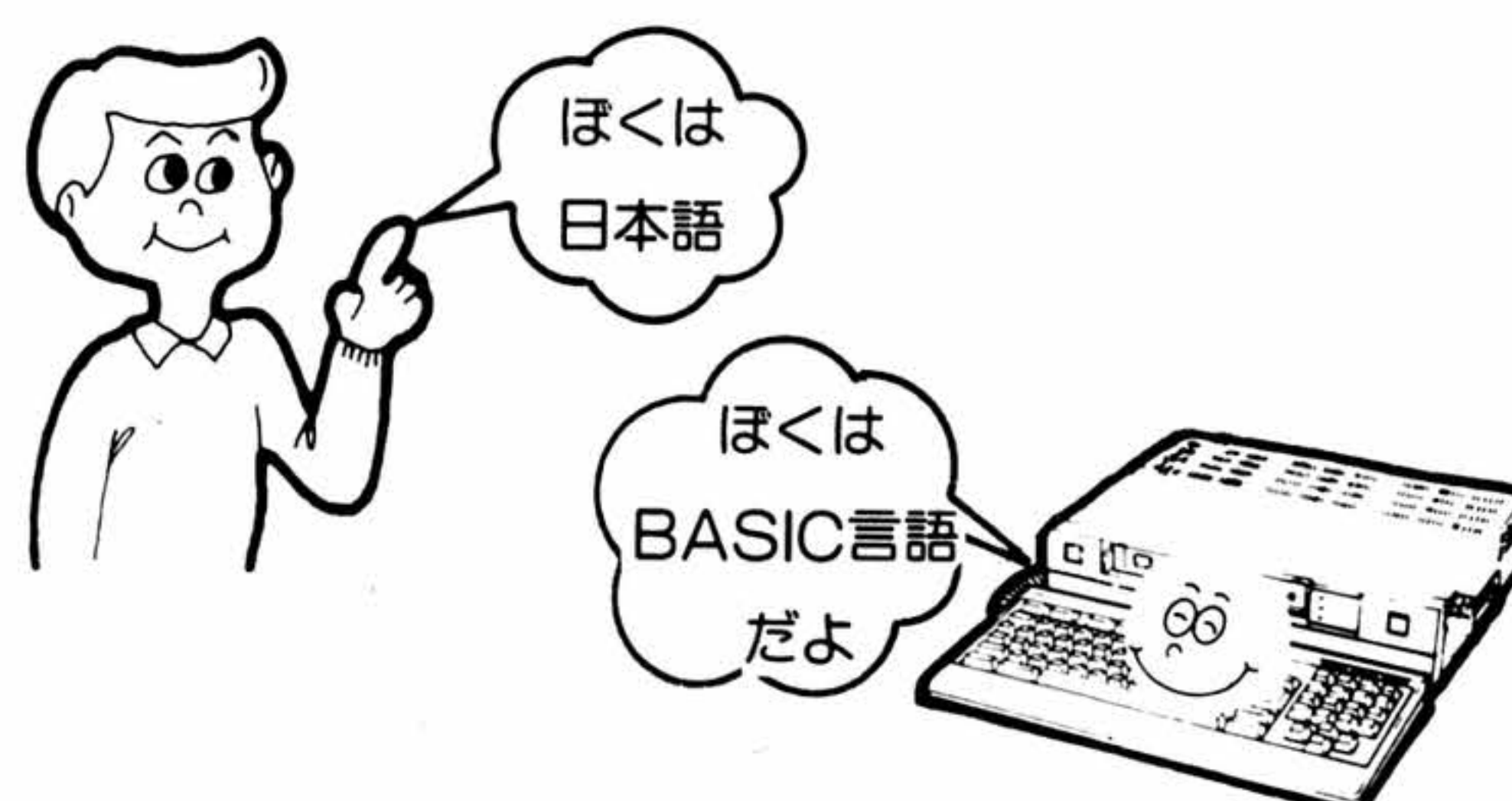
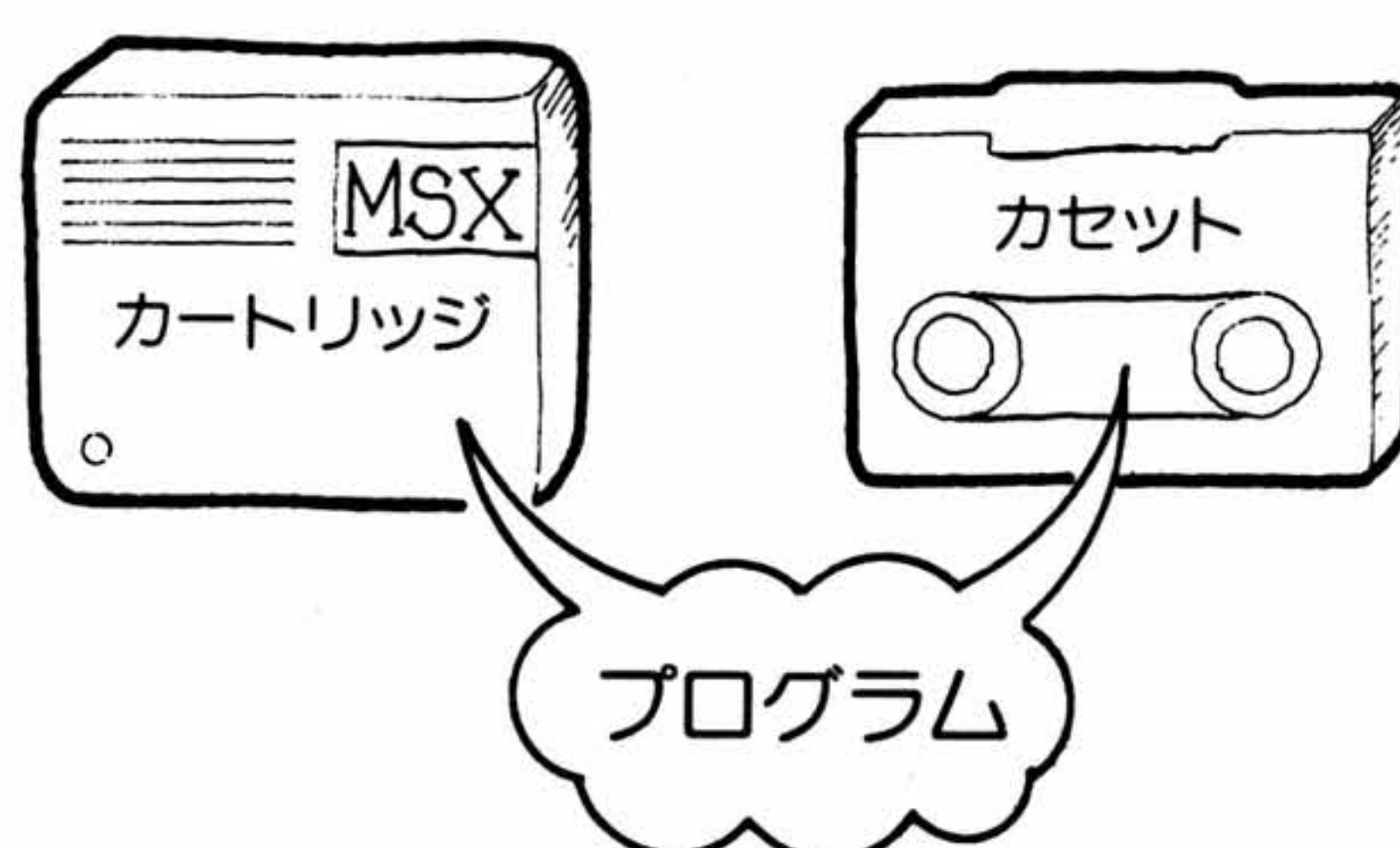
この章では、パソコンで何ができるのか、プログラムって何なのかを、はじめて使う人にもわかりやすいように説明してあります。
この章、および他の章も参考にして、少しでもパソコンと楽しくつき合えるようにしてください。

1. 1 とにかくさわってみよう!

1. 1. 1. パソコンでは何ができるの?

ゲームカートリッジを使ったことがありますか?
きれいな絵が出てきたり、音楽や車の音が出てきますね。そしてゲームの得点(スコア)が画面に出るでしょ。
パソコンではゲームができますね。でも、ゲームだけでしょうか? いいえ、パソコンはゲームのほかにも、算数の勉強に使ったり、お母さんの家計簿、家の電話帳の代りにもなります。
だけど、どうすれば勉強に使うことができるのでしょう。パソコンを買ったお店へ行って、勉強に使うカートリッジを探しますか? でも、なかったらどうしましょう。こんなときは、自分でプログラムというものを作って、パソコンに命令してやればいいんですよ。

プログラム? プログラムって何でしょう。その答えは後で説明しますから、その前にパソコンに命令する方法を覚えましょう。
私たちが、ほかの人に何か話すときは日本語で話しますね。外国人なら英語、フランス語、ドイツ語……といろいろあります。実は、パソコンにも決った言葉があります。それは、BASIC(ベーシック)言語というものです。(パソコン語ではありませんよ。)

初めからむつかしい言葉がでてきましたが、パソコンに間違った命令をしても、パソコンがこわれる心配はありません。そのかわり、「命令が違うよ。」という合図として「ピッ!」と音が出ます。
だけど、キーボードになれることが大切だから、パソコンの「ピッ」を気にしないで、キーを打つ練習をしましょう。(パソコンになれない間は、別冊の取扱説明書もいっしょに読んでね。)

用　語

■BASIC (ベーシック)
英語を基本とした初心者向けのプログラム言語です。BASIC言語もパソコンの機種によって少しづつ違いますが、このパソコンはMSX(エムエスエツクス) BASICという言葉を使います。

Beginners(ビギナーズ) All-purpose(オールパーパス) Symbolic(シンボリック) Instruction(インストラクション) Code(コード)
(初心者の)　(万能の)　(記号の)　(命令)　(符号)

1. 1. 2 簡単な命令をしてみよう

テレビとパソコンの電源を“入(ON)”にすると、図のようなメッセージが画面に出てきますね。(ROMカートリッジや周辺機器は、ちょっと外しておいてね!)
そして、「OK」という文字の下に白い四角形があるでしょう。これを**カーソル**といいます。
これからプログラムの練習をするとき、なにかキーを押すとこのカーソルの位置に文字が表示されます。

```
MSX BASIC version 1.0
Copyright 1983 by Microsoft
xxxxx Bytes free
Ok
□ ←カーソル

color auto goto list run
```

(パソコンの機種によって表示が一部異なります。)

それでは、画面に文字を表示するPRINT(プリント)という命令を使ってパソコンに命令を伝えてみましょう。まず、文字を見やすくするために、[CAPS] キーを押して大文字にしてください。そして、次のようにキーボードのキーを押してね。

(CAPSランプを点灯させます。)

[P][R][I][N][T]["][M][S][X]["]

画面に右の図のように表示されているはずです。
でも、これだけではパソコンは何もしてくれません。パソコンは、押したキーの内容を表示しているだけで、どうするのかわからないからです。
では、パソコンに命令を伝える [リターン] キーを押してください。

```
Ok
PRINT"MSX"□
          └カーソル
```

[リターン] キーを押すと

すると、画面に「MSX」という文字が出ますね。

PRINT "MSX"

は「" "で囲まれている文字を画面に表示しなさい。」という命令で、[リターン] キーを押すとその命令を実行します。だから「MSX」という文字が出たんだよ。そして、その下の「**OK**」は「次の命令はなんですか?」という意味です。

```
Ok
PRINT"MSX" ←命令
MSX        ←結果
Ok         ←「次の命令はなに?」
□            という意味です。
```

ちょっと一言

■初めの画面について
パソコンの電源を“入(ON)”にしたときに、初期画面という画面が出ます。パソコンにカートリッジや周辺機器をつけていると図と異なった画面になることがあります。
また、MSX2のパソコンでは、右の絵のような画面になります。

(MSX2のパソコンで、フロッピーディスクをつないだとき)

```
MSX BASIC version 2.0
Copyright 1985 by Microsoft
XXXXX Bytes free
Disk BASIC version 1.0
Ok
□←カーソル

color auto goto list run
```

今度は、
PRRINT"MSX"リターン

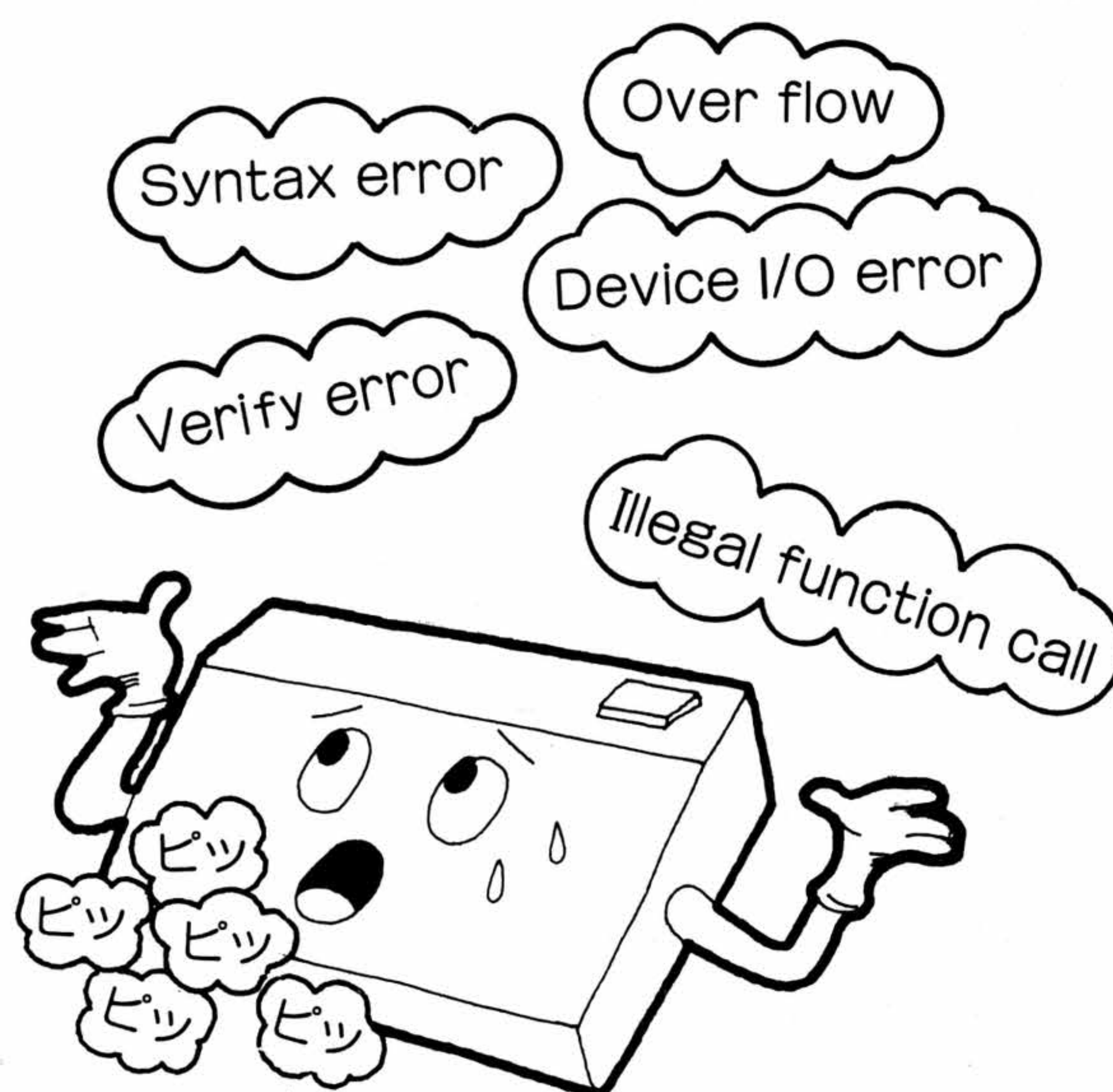
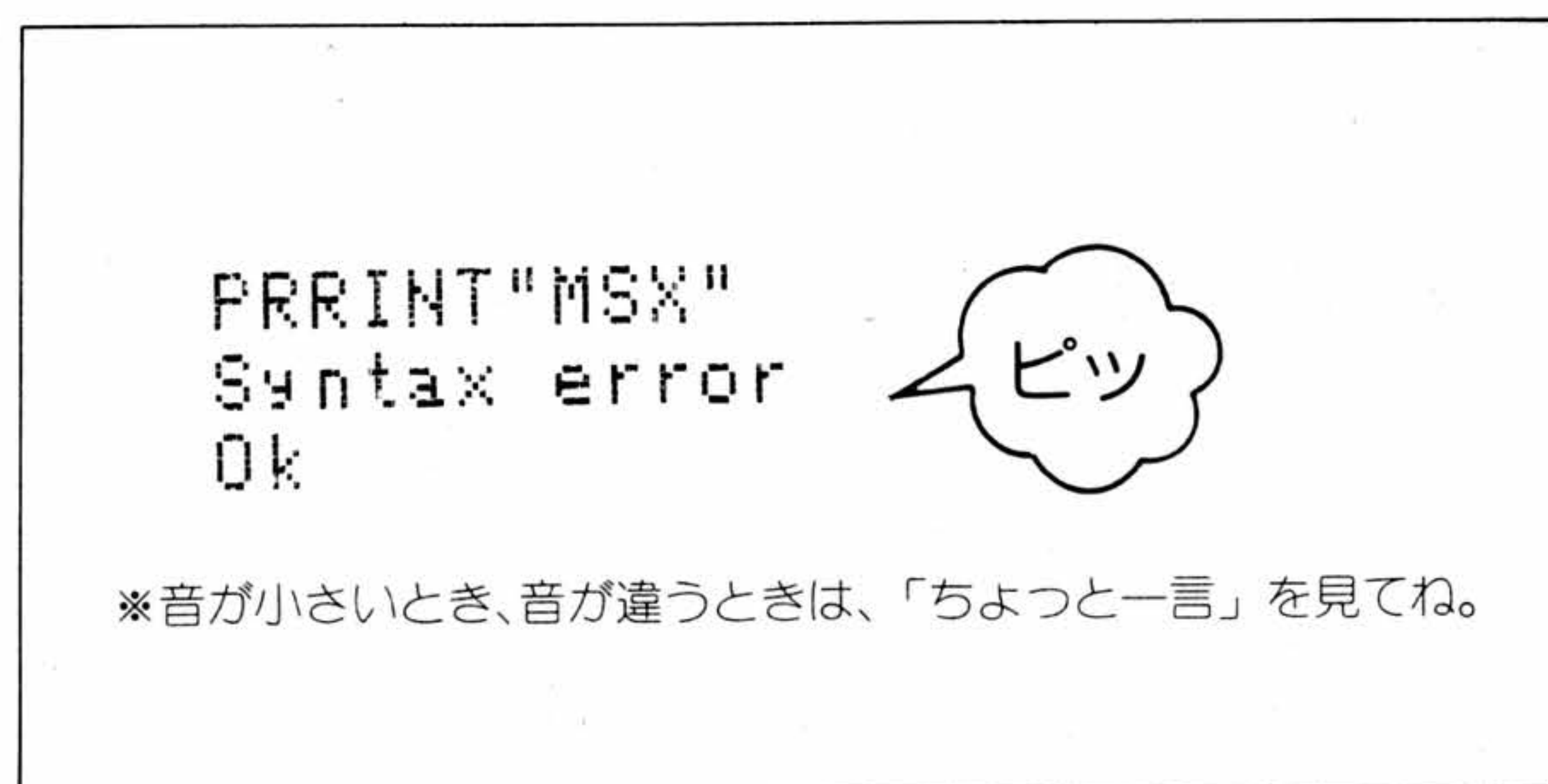
とキーを押してください。
すると、「ピッ」という音がして、画面には

Syntax error（シンタックス エラー）

と出ましたね。
これは、PRRINTという命令がないので、パソコンが「なにをするのかわかりません。命令が間違っていますよ。」という意味で、「ピッ」という音といっしょに間違いの種類を**エラーメッセージ**（今はSyntax error）できみにしらせているんです。

パソコンには決った命令（MSX BASIC）を正しくキーで打たないと動かないことがわかったかな?
でも、命令をまちがえたぐらいではパソコンはこわれないから心配ないよ（ピッ!ピッ!という音はうるさいかもしれないけど）。

```
PRRINT"MSX"
Syntax error
Ok
```


※音が小さいとき、音が違うときは、「ちょっと一言」を見てね。

ちょっと一言

■エラーメッセージの音
Syntax errorのときの音が、ちゃんと出ましたか?
音が聞えなかったときは、テレビの音量を大きくしてみてください。
また、MSX2ではエラーメッセージの音（音量、音色）を変えることもでき、エラーのときにチャイムの音を出したり、大きな音を出すこともできます。詳しくは第3章のSET BEEP命令を見てください。

■エラーメッセージの種類
命令が間違っているときに出るエラーメッセージにはいろんな種類があります。
どんな意味があるのかは、第5章で説明しています。

■大文字と小文字の命令
命令は、大文字でキーを打っても小文字で打っても同じように働きます。
しかし、命令のなかにほかの文字や空白を入れると間違いとなります。

PriNT
Print
} 働く

PR INT 働かない

1. 1. 3 PRINT命令で計算してみよう

PRINT命令は、文字を出すだけでなく計算もできます。次のようにキーを打ってください。

PRINT 5+2 [リターン]

```
PRINT 5+2
 7
Ok
PRINT 5-2
 3
Ok
PRINT 5*2
 10
Ok
PRINT 5/2
 2.5
Ok
```

どうなりましたか?。すぐ下に5＋2の答「7」が表示されたでしょ。ひき算、かけ算、わり算もできますよ。

ひき算は……PRINT　5－2 [リターン]

かけ算は……PRINT　5＊2 [リターン]

(かけ算の記号×の代りに＊(アスタリスク)を使ってね。)

わり算は……PRINT　5／2 [リターン]

(わり算の記号÷の代りに／(スラッシュ)を使ってね。)

それぞれ順にキーを打つと、図のように答がでます

1. 1. 4 多くの命令を一度にする

今までは一つのPRINT命令でパソコンを使ってみましたが、たくさんの命令を一度に実行させるにはどうしたらいいでしょう。
一つの方法としては、いくつもの命令をつづけてキー入力します。次の命令をキー入力してください。

PRINT　5＊2：PRINT 5/2 [リターン]

(：のコロンで命令をつなぐことができます。)

```
PRINT 5*2:PRINT 5/2
 10
 2.5
Ok
```

どうです? 5＊2の答を表示した後で、5÷2の答を表示したでしょ。
このように命令をつづけていくこともできますが、この方法ではゲームなど複雑なことはできません。
そこで、次の1. 2項で説明するプログラムが必要になってきます。

ちょっと一言

■計算するときに気をつけて (第5章参照)
- このパソコンでは、数字が14桁まで表示できます。もっと大きな数字や、小数点より14桁以上小さな数字は指数表示されます。
- 計算式の中に(　)や^，＊,／の記号があると計算の順序が変りますよ。

■PRINT命令の使い方
- PRINT "5＋2" のように計算式を" "で囲むと計算はできません。5＋2と式のまま表示されます。
- PRINTの代りにクエスチョンマーク(?)を使うことができます。?5＋2をためしてみてください。

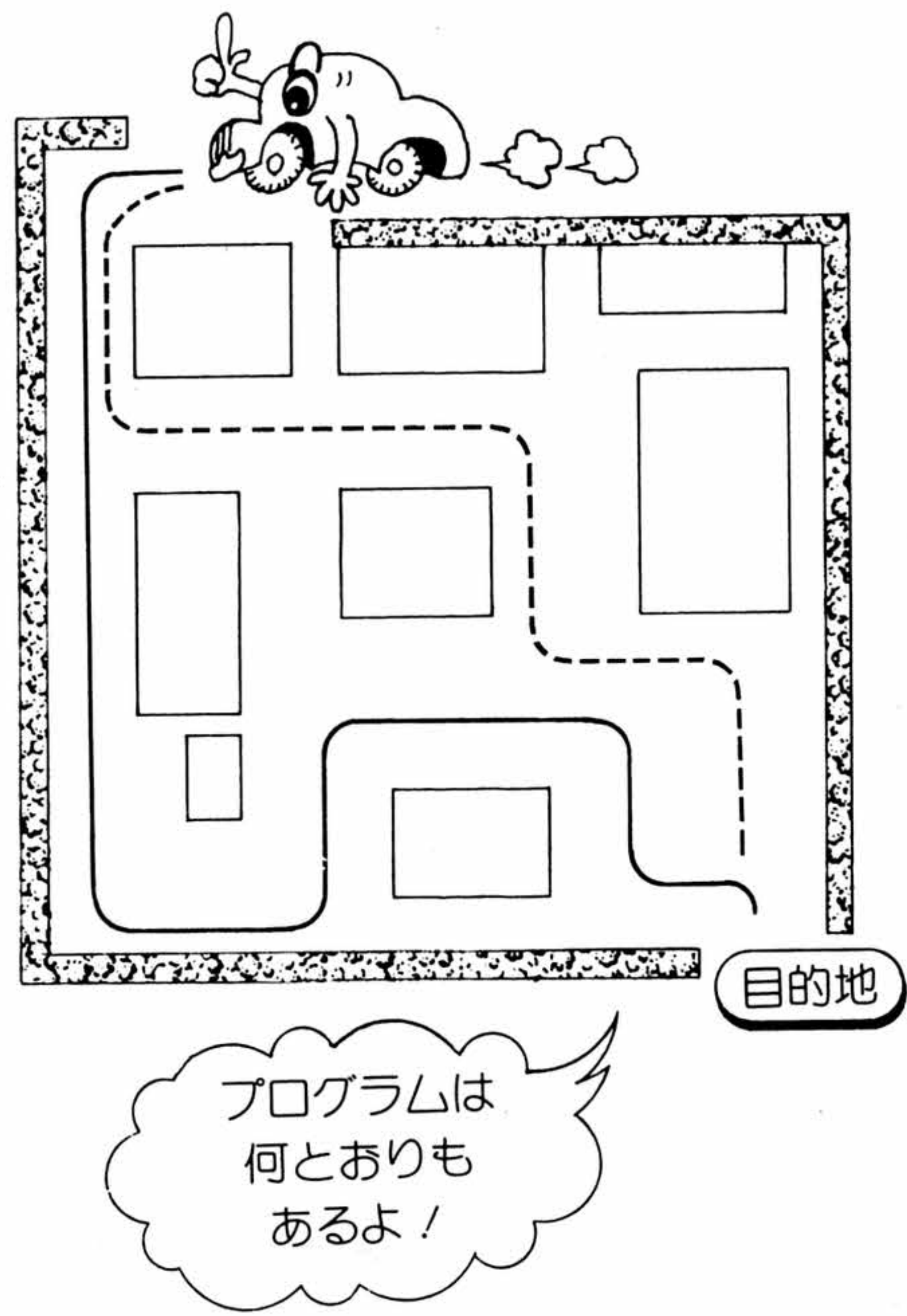
(?をプログラム中で使用したとき、リストアウト表示のときには?がPRINTとなっています。しかし、LPRINT命令の代りにL?を使用することはできません。(LPRINTと表示されますが、変数LにPRINT命令がついたものと見なされます。))

1．2 プログラムを作るための基礎知識

1．2．1 プログラムってどんなもの?

ゲームカートリッジをパソコンにつないだとき、必ずゲームのタイトルとゲームの遊び方の説明が表示され、ゲームが始まりますね。パソコンの電源を切って、入れなおしても必ず同じように始まります。これはゲームカートリッジの中にどんな順序で画面を表示するのかを、たくさんの命令を組合せて記憶しているからです。このように、多くの命令を組合せて、順序よく並べたものをプログラムといいます。

右の絵は、車が目的地までに進むときの道順が何とおりもあることを表わしています。プログラムも同じように、MSX BASICの書き方に従っていれば、どういう順序で命令してもいいんです。あなただけのオリジナルプログラムを作ることができます。

ただし、交通違反があると目的地まで行けませんよ（プログラムにエラーがあると思ったとおりには動きません）。

1．2．2 プログラムを作る前に

プログラムを作るには、次のことを覚えておきましょう。

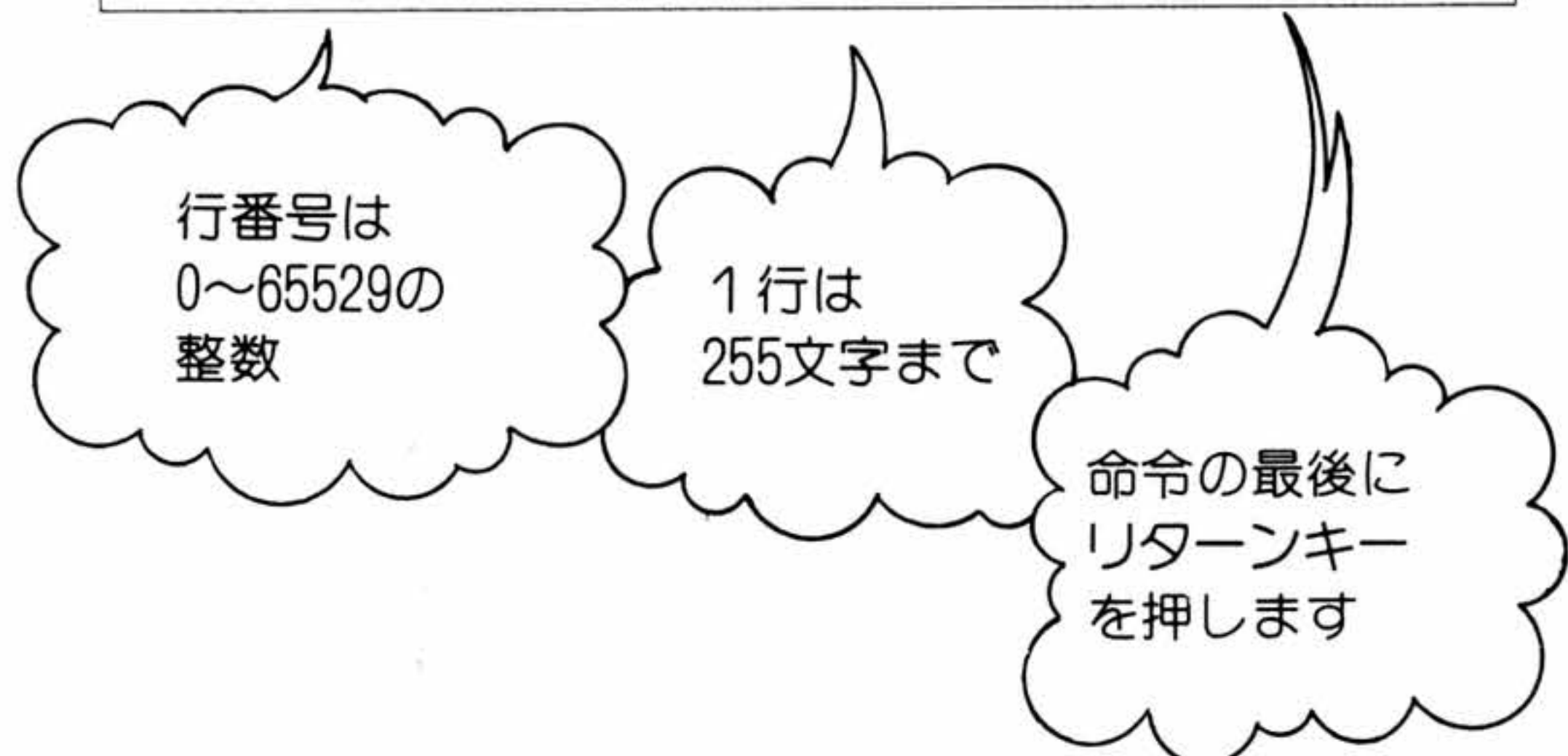
■プログラムには行番号がいるよ

「プログラムはたくさんの命令を並べたもの」といいましたが、パソコンにはっきり順序がわかるようにするため、各命令に番号をつけてやります（行番号といいます）。

1．1項の計算式をプログラムにしてみましょう。

```
100  PRINT  5＋2 [リターン]
110  PRINT  5－2 [リターン]
120  PRINT  5＊2 [リターン]
130  PRINT  5／2 [リターン]
140  END [リターン]
```

正しく入力できましたか?
行番号100の命令をすべて入力しても、結果も「OK」も表示されませんね。110～140も同じです。
これは命令の前に番号を入れているためで、こうするとパソコンは「プログラムを作っている」と判断してパソコンの内部に記憶します（計算もしません）。

〈プログラムの入力〉

プログラムを実行させるには、全部の行番号の命令をパソコンに記憶させて（プログラムを全部入力して）から、「プログラムを実行しなさい」というRUN命令をキー入力します。
それでは実行させましょう

RUN [リターン]

画面は図のようになりましたか？
4つの計算が一度に実行され、結果が表示されましたね。このようにプログラムを作ると一度にたくさんの命令を実行できます。
もし、エラーメッセージが出たり、違った結果になったら、プログラムを見直してください。そして1．2．3の「プログラムの修正と追加のしかた」の項をみて、修正しましょう。

〈プログラムの実行〉

RUN [リターン]

さあ、実行しよう

```
run [リターン]
 7
 3
 10
 2.5
Ok
```

結果

■プログラムは表示を消しても残っている

プログラムとして作った命令は、電源を切ったり、リセットボタンを押さないかぎりパソコンはちゃんと覚えていますよ。
画面には、今までキー入力した文字が表示されていますね。一度画面の文字をきれいに消してしまいましょう。

[SHIFT] キーを押しながら[CLS HOME] キー

を押してください。「せっかく入力したプログラムが消えてしまった。」と言わないでね。画面は消えてもパソコンはプログラムを覚えていますよ。

LIST [リターン]

とキーを押してください。画面に行番号100から140までのプログラムが表示され、あとの文字は表示されないでしょ！
つまり、パソコンはプログラムを覚えているけど行番号のない命令は消えてしまうんです。

LIST 始めの行番号－終りの行番号

この行番号から表示します

"－"はマイナス記号です

この行番号まで表示します

ちょっと一言

■プログラムが消えるとき

プログラムといえども、電源を切ったりリセットボタンを押すと消えてしまいます。
大切なプログラムはカセットテープなどに記録しましょう。
また、次の命令をキー入力しても消えます（くわしい説明は第3章を見てね）。

NEW [リターン]

覚えておいてね

■行番号を自動的に作る……AUTO命令

100，110という行番号をいちいちキー入力しなくても、パソコンが自動的に行番号を作ります。くわしい使い方は第３章を見てください。

AUTO〔〈開始する行番号〉〕〔,〈増分〉〕

始まりの行番号　　行の間隔

※〔 〕内は省略できます。
〔 〕内を全て省略すると、開始する行番号と増分に10が設定されます。

■行番号のつけ方

1．行番号は、０から65529までの整数をつけます。
2．小さい行番号から実行します。
3．ふだんは、10，20，30など10番おきにつけます。こうすると、プログラムをあとから追加するときに便利です。
4．AUTO命令を使うと便利です。

■ファンクション・キーの使い方

LIST，AUTOなどの命令はよく使うので、キーボードのファンクション・キーに命令が入っています。つまり、キーを１回押すだけで命令の文字がキー入力できるわけ。ためしに F4 キーを押してみて。ちゃんとlistが出たでしょ。

F2……AUTO命令
F4……LIST命令
F5……RUN命令と リターン キー

F1	color
F2	auto
F3	goto
F4	list
F5	run
F6	color 15, 4, 7
F7	cload"
F8	cont
F9	list.
F10	run

↙MSX2+では load"です。

■ダイレクトモードとプログラムモード

一度入れたプログラムは、RUN リターン とキーを押せば、何回でも結果を表示します。この方法を**プログラムモード**（プログラムによる実行）といいます。

この反対に、PRINT　５＋２のように行番号のない命令は リターン キーを押すとすぐに実行します。この方法をダイレクトモード（直接実行）といいます。

ダイレクトモードではすぐに命令の結果がわかるけど、プログラムとして残らないので画面を消すともう一度キー入力しなければなりません。そして多くの命令を入れることはできません。プログラムは、命令の結果がすぐにはわからないけど、パソコンが命令を記憶しているから、何回も繰り返して実行できます。

なにかむつかしい呼び方が出てきたけど、これはプログラムの説明のときに使うことばだから、呼び方を間違えても、パソコンはちゃんと動くよ。

ダイレクトモードだよ

```
PRINT  2*7  リターン
```

```
10 PRINT  2*7
RUN      リターン
```

プログラムモードだよ

1．2．3　プログラムの修正と追加のしかた

プログラムを作っている途中で、キーを押しまちがえたり新しい命令を追加したいとか、プログラムの修正・追加は必ずあります。この項ではその方法を覚えましょう。

■修正するプログラムを画面に表示する。

キー入力したプログラムを修正するときは、まずプログラムを画面に表示します。LIST命令でプログラムを出しましょう。

●全部のプログラムを表示するときは

LIST [リターン]

●プログラムの一部だけを表示するときは

LIST100 [リターン]

└表示したい行番号を入れる。(例．行番号100)

■修正の仕方

今までに作ったプログラムを修正してみましょう。

●カーソルを動かして文字を変えるとき、

たとえば行番号100のPRINT 5＋2の5を6に変えてみましょう。

①カーソルを「5」のところへ動かします。

② [6] キーを押すと、5が6に変ります。

③ [リターン] キーを押します。

文字を修正してから、かならず [リターン] キーを押してね。

[リターン] キーを押さないと、画面の表示が変わるだけで、パソコンはプログラムが変わったことを覚えてくれません。

●ある行番号の命令をすべて変えるとき

たとえば行番号110の命令をPLAY "CDE" と変えるときは、

110　PLAY "CDE" [リターン] (PLAY"CDE"はドレミの音を出す命令です)

と、この行番号の命令をはじめから打ち直します。

すると、

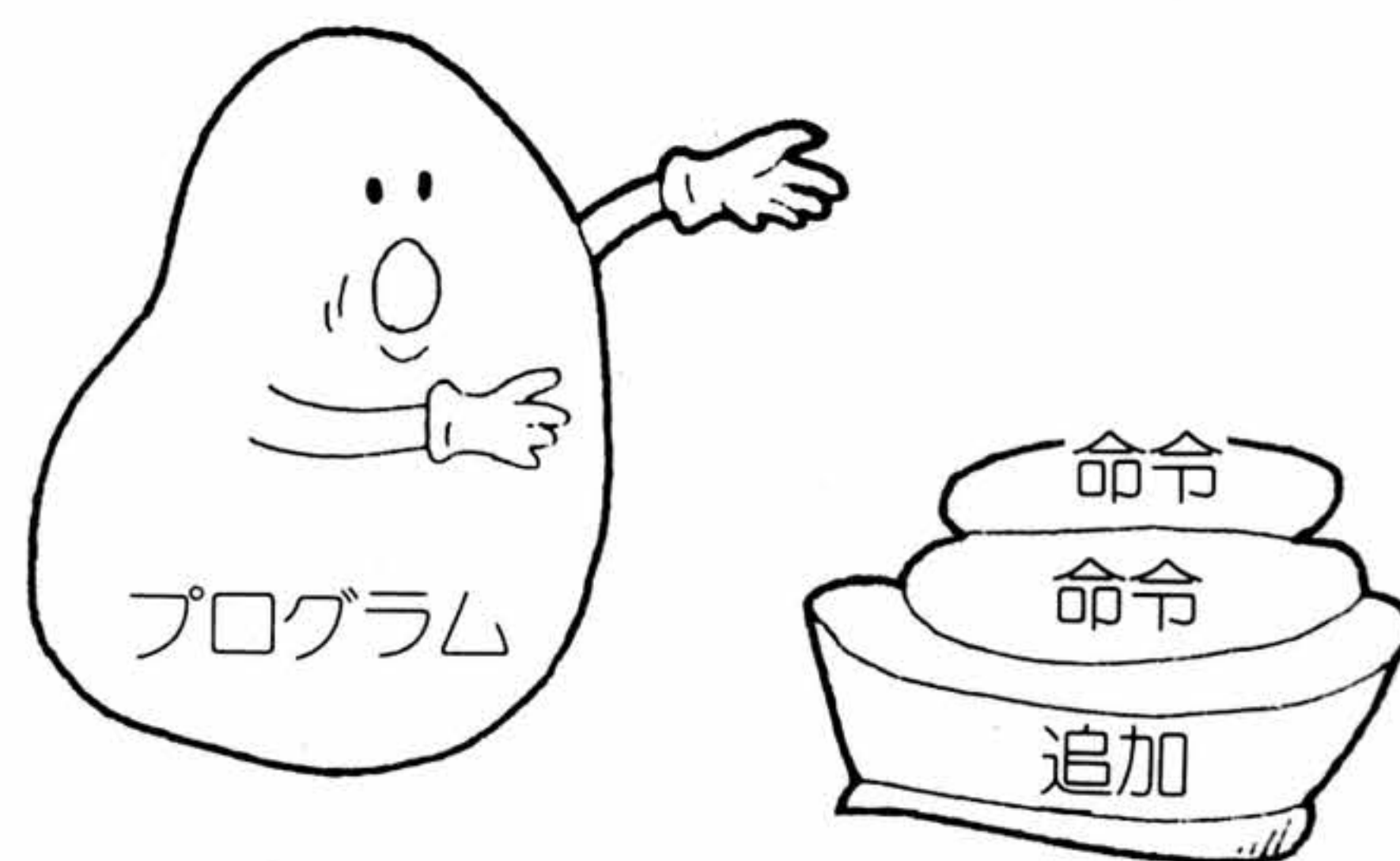
今まで入っていた行番号110の命令はすべて変ります。

●命令を追加するとき

たとえば行番号130と140の命令の間に、命令を追加したいときは、次のようにします。

135　PRINT　6^2 [リターン] (^(ベキ)は、6^2の計算をします)

これで、130と140の間に行番号135の命令が入りました。ためしにLIST命令でプログラムを表示してみてください。

プログラム　命令　命令　追加

プログラムに間違いがあるとエラーメッセージと行番号が表示されます。その行番号を指定すると早く修正できるよ。

```
run
Syntax error in 120
Ok
list 120
```

カーソルをここへもってくる

```
100 PRINT 5+2
```

↓ [6] キーを押す

```
100 PRINT 6+2
```

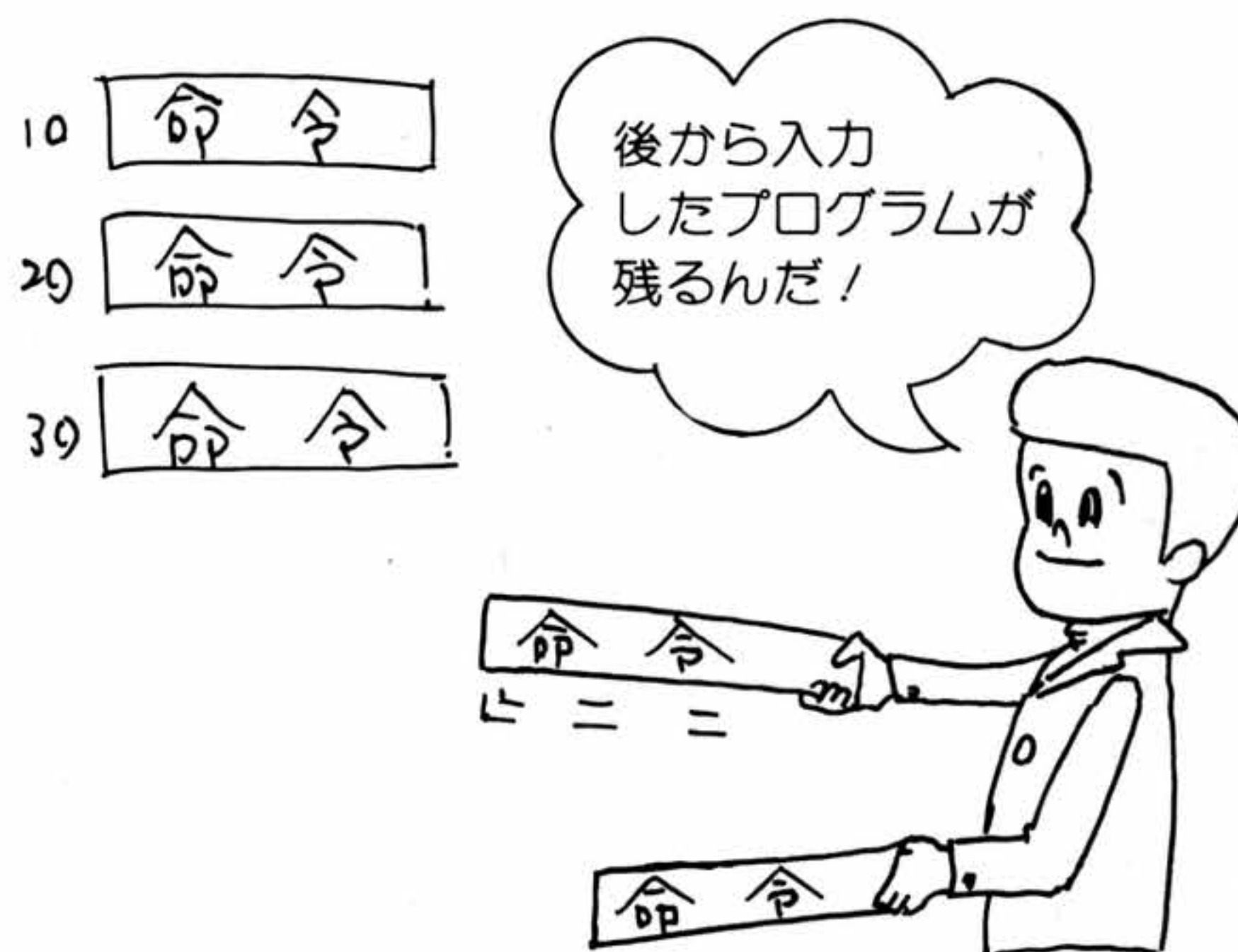
↓ [リターン] キーを押す

修正終り

忘れずに！

[リターン]

10 命令
20 命令
30 命令

後から入力したプログラムが残るんだ！

プログラムの修正が終ったら

修正がいくつもあると、[リターン] キーを押し忘れることがあります。修正が終ったら、LIST [リターン] として、プログラムが正しく修正されているか確かめてね！

●ある行番号の命令を消すとき

たとえば、さっき追加した行番号135の命令を消したいとき、

135 リターン

とします。これは、「行番号135に新しいプログラムを入れたけど、命令は何もないよ。」ということです。

行番号　リターン

1行分のプログラムを、消します

プログラムの修正の方法はこのほかにもいろいろあります。取扱説明書や第3章も読みましょう。

1．2．4　変数を使おう

今までの説明で、計算のプログラムの作り方がわかりましたね。でも、この方法だと計算する数字がいくつもあるときはいちいちプログラムの数字を直さなければなりませんね。

そこで、変数を使う方法を覚えましょう。

いままでに入れたプログラムを右の図のように追加・修正してください。

（リターン キーを忘れないでね！修正が終ったら、一度画面の表示を消してから、LIST してね。）

```
10 A=5
20 B=2
100 PRINT A+B
110 PRINT A-B
120 PRINT A*B
130 PRINT A/B
140 END
```

10・20：プログラムを追加します。

100〜140：数字の5や2をA，Bに修正します。（PLAY "CDE" は消します。）

正しくプログラムが修正できたら、

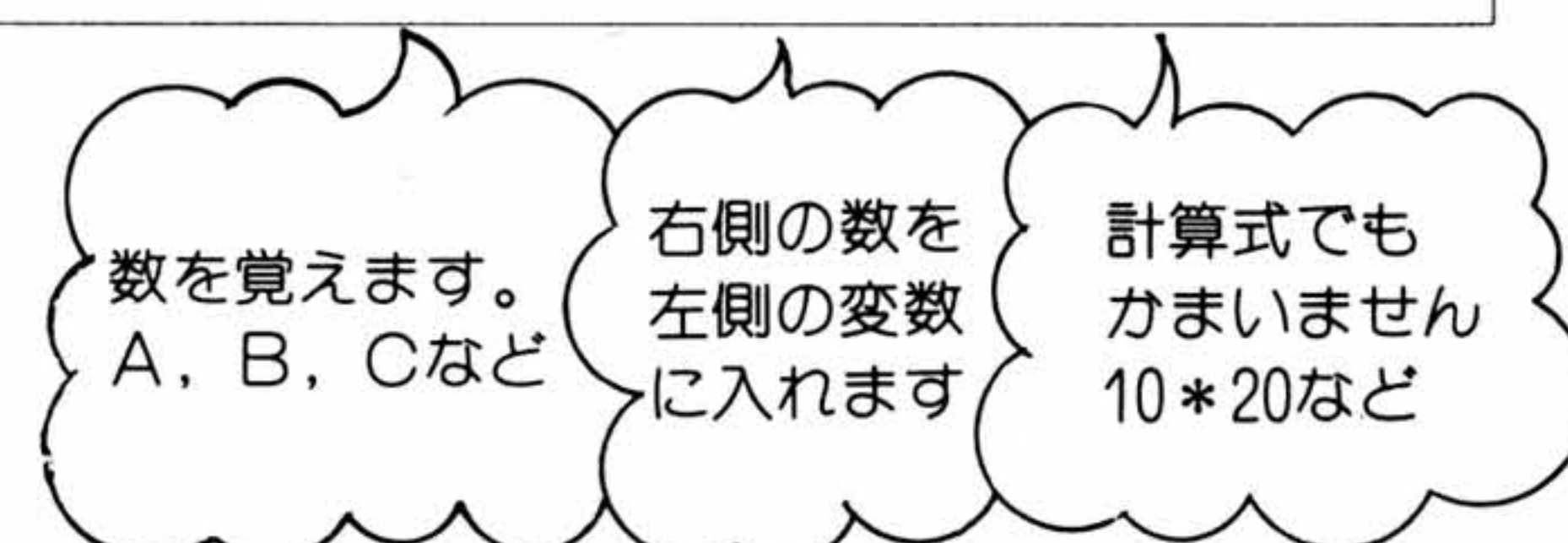
RUN リターン （または F5 キー）

してみましょう。どうです？。前と同じ計算結果が出ましたね。

変数：数を覚えています。数を入れておく箱のようなもの

行番号10と20を見てください。「A」「B」が変数を表わす記号です。この意味は、「変数のAに、5という数をいれなさい。」という意味で、「これからあとでてくるAは、数字の5だよ。」と言っています。だから100行から後ろのプログラムの中にあるAは、数字の5として計算されています。

変数名　＝　数

- 変数名：数を覚えます。A，B，Cなど
- ＝：右側の数を左側の変数に入れます
- 数：計算式でもかまいません　10＊20など

覚えておいてね

■変数として使える文字は、

1．変数は2文字まででです（2文字以上は無視されます）。
2．最初の一文字は英文字です。
3．第3章のベーシック用語は使えないよ（PRINTなど）。

良い例	悪い例
A，B	ABCD，ABC……どちらも変数ABとなる。
A1，A2	
AA，AP	APRINT……ベーシックの言葉が入っている。

■＝の意味

変数を使った計算式の中で使うイコール（＝）記号は、等しいという意味ではありません。＝の右側の数を左側の変数に入れる（代入する）という意味です。

例．C＝10＋20……10と20を加えた数をCに入れます。

　C＝C＋1…………変数Cに1を加えて、もう一度変数Cに入れます。

1．2．5　ベーシックにはこんな言葉があるよ

ベーシックで使う言葉は変数名に使ってはいけません。でも、どんな言葉があるのかな？
言葉は大きく分けると3種類あるよ。

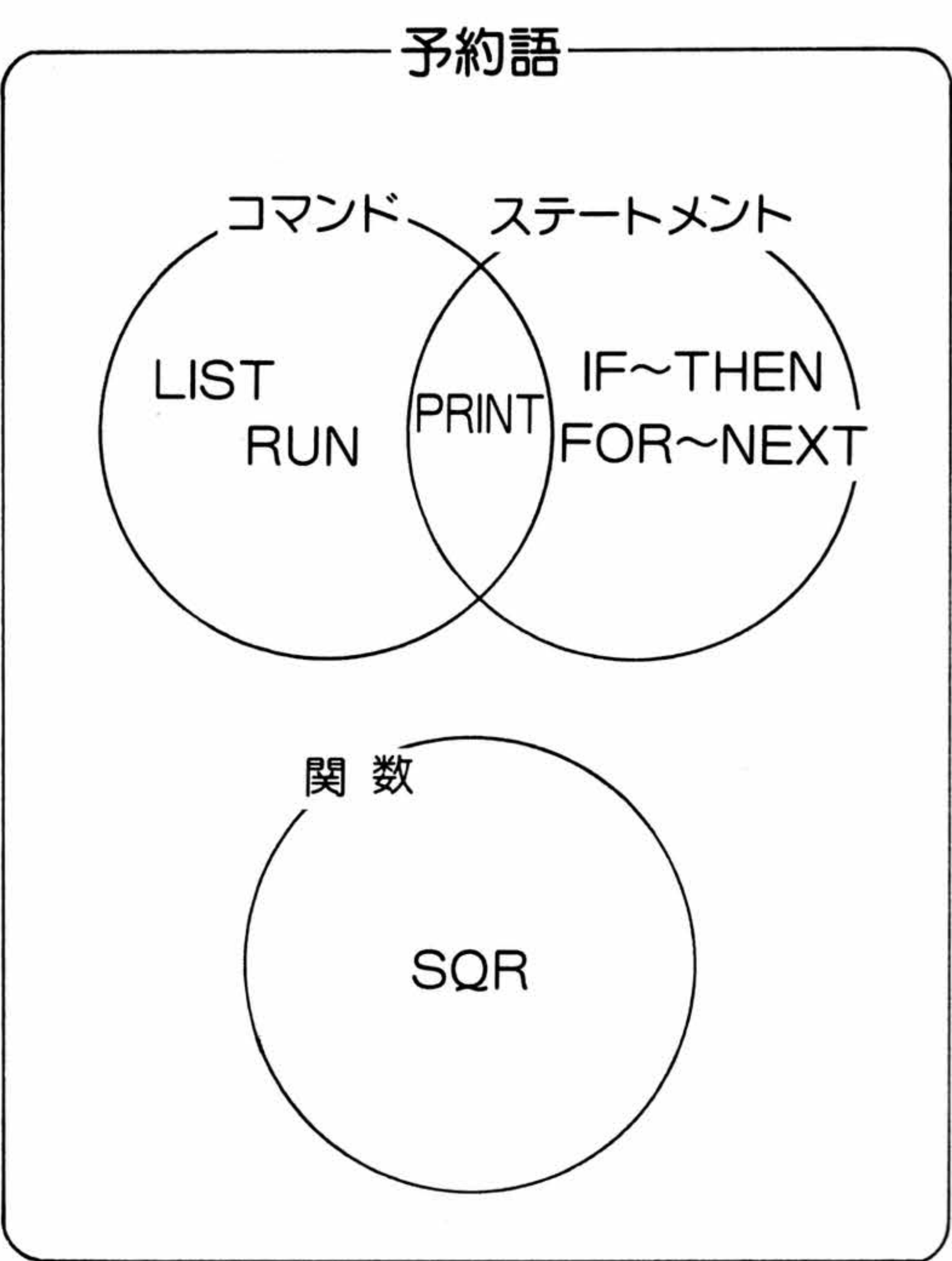
●**コマンド**…………プログラムを作ったり修正するときに使う命令です。
例．LIST，RUN

●**ステートメント**…プログラムの中で使う命令です。プログラムの修正や、チエックのときにも使います。
例．PRINT，GOTO

●**関数**………………ベーシックであらかじめ決められている計算の方法です。コマンドやステートメントと組み合わせて使います。
例．SQR，LOG

これらをまとめて予約語というんです。予約語は英語を基本に作ってあり、プログラムを作るときよく使います。第3章では、それぞれの命令の使い方を説明しています。また、第5章には予約語の一覧表がありますから見ておきましょう。

予約語

コマンド　ステートメント
LIST
RUN
PRINT
IF～THEN
FOR～NEXT
関数
SQR

正確に呼び方の区別が決っているわけではないので、こういう呼び方があるということを覚えていてください。

1．2．6　プログラムは、まとめよう

プログラムには行番号がいること、そして行番号の順に命令を実行していくことはわかりましたね。
では、１つの行番号には１つの命令しか入らないのでしょうか？　いいえ、命令を続けてもいいんです。
ただし、命令を続けて入れるには次のように**コロン**（：）で命令と命令を区切らなければなりません。

10　A＝5：B＝2 リターン

行番号20の命令を消して（20 リターン）、LISTしてみましょう。右の図のようになりましたか？

RUN リターン

すると8ページと同じ結果になります。

```
10 A=5:B=2  リターン
20 リターン
```

```
10 A=5:B=2
100 PRINT A+B
110 PRINT A-B
120 PRINT A*B
130 PRINT A/B
140 END
```

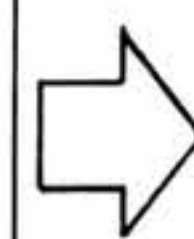
このように１つの行番号にいくつかの命令をつづけて入れてもいいのです。

１つの行番号に続けて命令を入れると何かいいことがあるのでしょうか？
それは、

- **プログラムの行番号が少なくなり、プログラムが短くなる。**
- **プログラムが見やすくなる。**

変数を10個使ったプログラムを考えてみてください。変数に数を入れるだけで10行（10〜100）もかかって見にくいものとなります。これを右図のようにするとすっきりするでしょ。
１つの行番号に、２つ以上の命令を入れたものを**マルチステートメント**と言います。

```
10 A=10
20 B=15
20 C=20
30 D=25
40 E=30
```

⇨

```
10 A=10：B=15：C=20
20 D=25：E=30：F=35
30 G=40：H=45
```

覚えておいてね

○１つの行番号には、行番号を含めて255文字まで命令が入る

○命令と命令の間はコロン（：）で区切る。

1．3 便利な命令

行番号と命令文によってプログラムが作れること、またプログラムの修正方法などの基礎知識が理解できましたか？
この項では、プログラムを作るときに知っていると便利な命令を紹介しましょう。おもしろいプログラムを作るときに、きっと役立ちますよ。

1．3．1 計算に便利な関数

予約語のなかに関数ということばがありましたね。
関数ってなんでしょう。

関数は、数値などのデータを与えると、それに対して計算や操作を行なう命令で、単独では用いられず必ず他の命令と組み合せて使います。

むずかしい説明はやめて、とにかく使ってみましょう。カンタンに使えるかもしれませんよ。

例としてSQRという関数を使ってみましょう。

PRINT SQR（4）[リターン]

```
PRINT SQR(4)
 2
Ok
```

と打ち込んでください。2と表示されましたね。
SQRという関数は、数の平方根を求めます。もし、SQRという関数を使わずに平方根を求めるとしたらたいへんなプログラムが必要になります。

プログラムを作るときに計算式が複雑になったら、何かカンタンな関数がないか調べてください。あらかじめパソコンが覚えている関数を使うと便利ですよ。

「単独では用いられない」という意味がわからない人は、

SQR（9）[リターン]

と入力してみてください。エラーとなりますね。これは、9の平方根をどうするのかわからないためのエラーなんです。
SQR（9）の前にPRINTを入れると、「SQR（9）の計算結果を画面に表示しなさい」という命令になるんだよ。

```
SQR(9)
Syntax error
Ok
PRINT SQR(9)
 3
Ok
```

1．3．2　数を自由に入力する……INPUT

1．2項で練習した変数（AとかB）に、プログラムを修正しないで数を入れられると便利ですね。こんなときには、INPUT（インプット）命令を使います。
次のようにプログラムを修正してください。

```
10 INPUT "A=";A
20 INPUT "B=";B
```

LISTすると右の図のようになります。

```
LIST
10 INPUT"A=";A
20 INPUT"B=";B
100 PRINT A+B
110 PRINT A-B
120 PRINT A*B
130 PRINT A/B
140 END
```

それではプログラムを実行してみましょう。すぐには結果が出てきませんね。そのかわり

A=? □

と表示されていますね。これは「変数Aには数字の何を入れますか？」とパソコンがきいているのです。
ためしに、

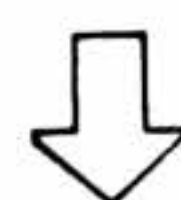
5 [リターン]

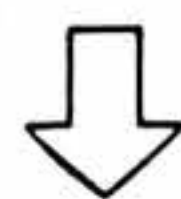
とキーを押してください。すると今度は、「変数Bには数字の何を入れますか？」ときいてきます。

B=? □

今度は、2と [リターン] キーを押してみてください。すると、すぐ下に計算結果が表示されます。

もう一度、RUN [リターン] とキーを押すと、またAの数を何にするかきいてきます。
このようにAとBの数は自由に変えることができるので、何回も計算することができます。

```
RUN
A=?
```

5 [リターン] と押すと

```
RUN
A=? 5
B=?
```

2 [リターン] と押すと

```
RUN
A=? 5
B=? 2
 7
 3
 10
 2.5
Ok
```

キー入力（A=? 5、B=? 2）

計算の結果を表示します。（7、3、10、2.5）

ちょっと一言

■INPUT，PRINT文の見出し

INPUT命令の" "で囲んだ文字はそのまま画面に出てきましたね。これは**プロンプト文**と言って、キー入力する数の意味をわかりやすくするために入れます（何も入れずに、INPUT Aとしても同じように計算はします）。これと同じように計算結果もわかりやすくするために、右の図のようにすることもできます。
セミコロン（；）については、第3章の「PRINT」を読んでください。

```
10 INPUT "A=";A
20 INPUT "B=";B
100 PRINT"A+B=";A+B
110 PRINT"A-B=";A-B
120 PRINT"A*B=";A*B
130 PRINT"A/B=";A/B
140 END
```

1・3・3　プログラムの流れを変える命令

プログラムは、行番号の小さいものから実行されていくけど、GOTO，GOSUB命令を使うと別の行番号から実行させることができます。

```
10 INPUT "A=";A
20 INPUT "B=";B
100 PRINT A+B
110 PRINT A-B
120 PRINT A*B
130 PRINT A/B
140 GOTO 10
```

■GOTO（ゴーツー）命令

プログラムの実行を指定した行番号へ移します。この命令で同じプログラムを何度も繰り返すこともできるよ。今まで入力しているプログラムの行番号140を次のように変えてみてください。

　140　GOTO　10

一度、RUN [リターン] とキーを押すと変数AとBの数をたずねてきます。そしてAとBの数を入力すると、計算が行なわれますね。
だけど、今度は計算の後に「OK」は表示されず、またAの数をたずねてきます。そうです。このGOTO命令を使うと何度もAとBの数を入れなおして計算できるのです。

```
run
A=? 6
B=? 3
 9
 3
 18
 2
A=?
```

つまり、今までは130，140行とプログラムを実行して終了（ダイレクトモードにもどる）していましたが、今度は行番号140のGOTO命令で再び10行へプログラムの実行が移っています（プログラムモードが続いている）。

GOTO　　行番号

行番号のところにある命令を
実行しなさい

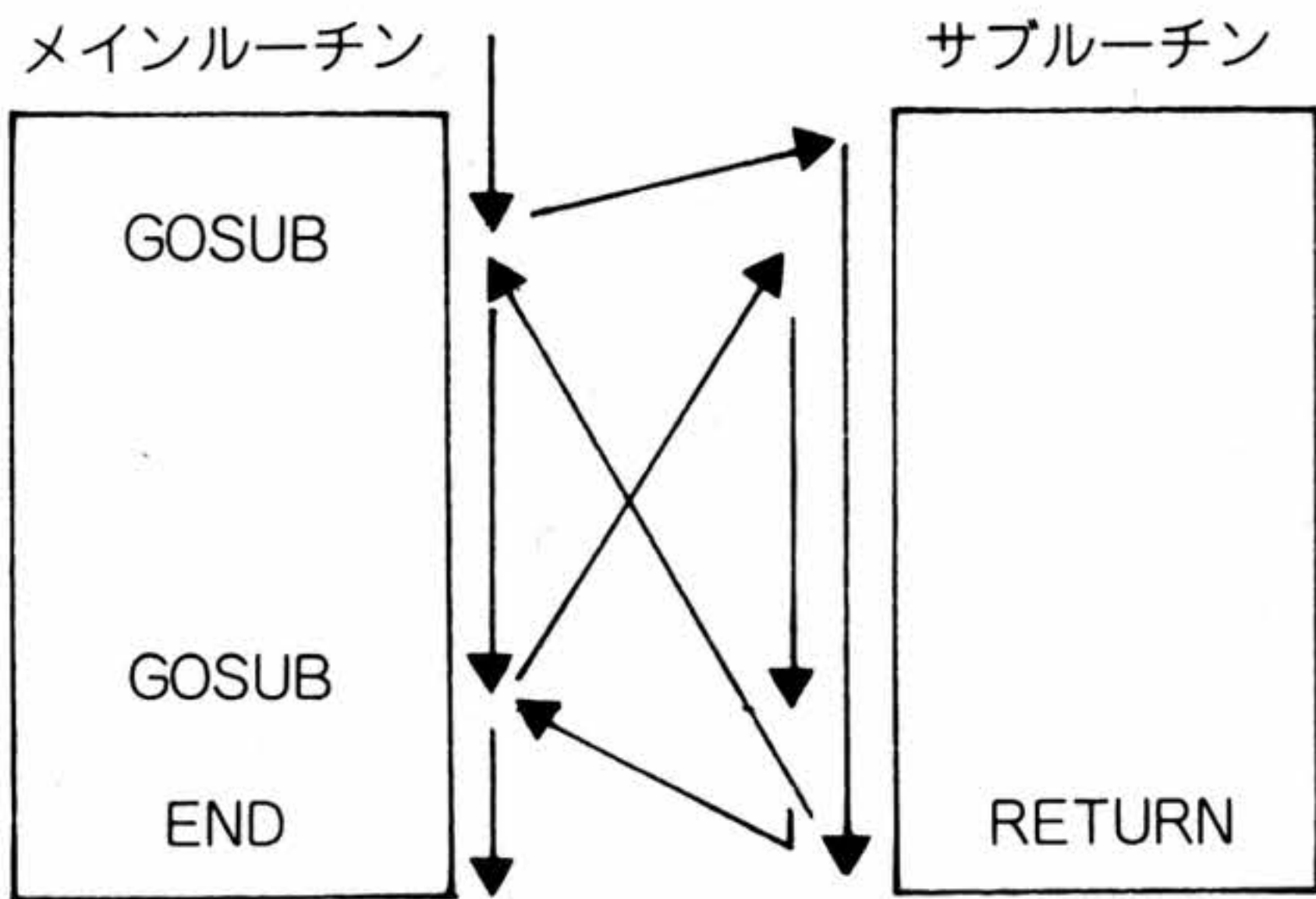
[CTRL] キーを押しながら [STOP] キーを押すとプログラムが止まります。

[CTRL] キーと [STOP] キー

■GOSUB（ゴーサブ）～RETURN（リターン）命令

プログラムの中で何度も同じ処理（計算など）をさせるときは、GOSUB命令を使うとプログラムが短くできます。

GOSUB命令によって実行を移すプログラムを"**サブルーチン**"といいます。サブルーチンの最後には必ずRETURN（リターン）命令があり、GOSUB命令の次の命令を実行します。（もとのプログラムの流れにもどります。）
もとのプログラムの流れを、サブルーチンに対して"**メインルーチン**"といいます。（第2章、第3章参照）

メインルーチン
GOSUB
GOSUB
END
サブルーチン
RETURN

ちょっと一言

■プログラムの流れを変える（ジャンプ）
GOTO，GOSUB命令などによってプログラムの流れを変えることをジャンプといいます。流れを変える命令には、このほかに割込みなどがあります。（1．5．8 参照）

1・3・4 もし～だったら……IF～THEN～ELSE

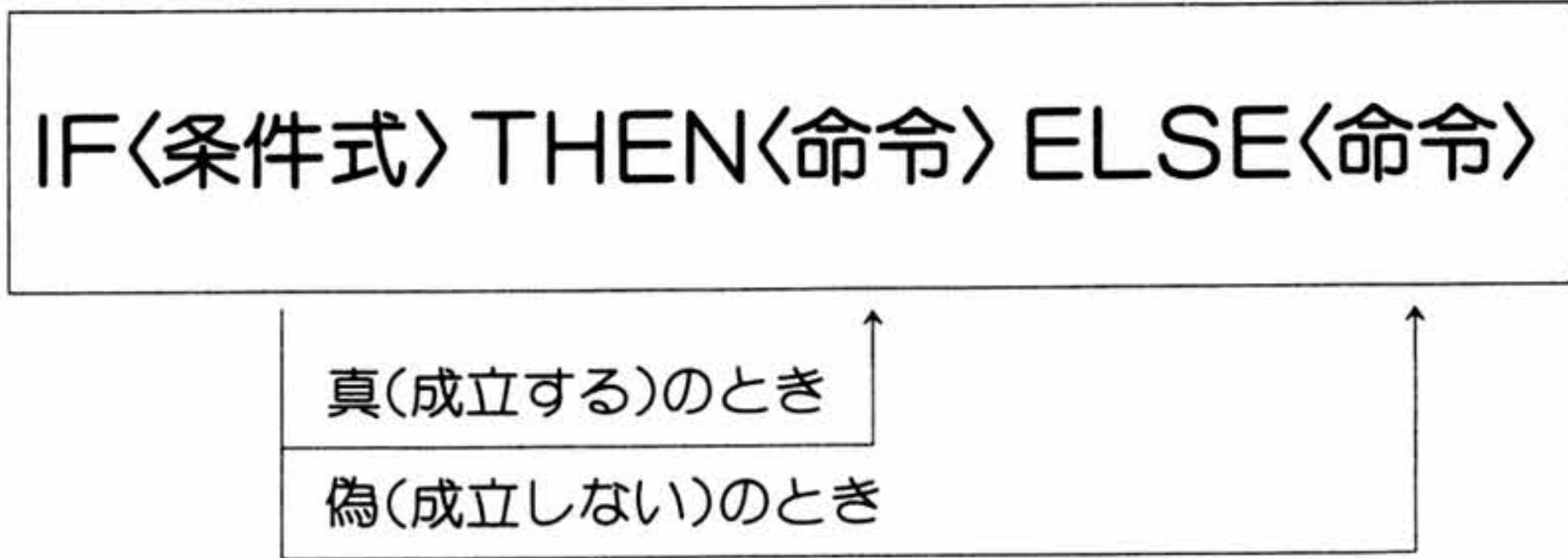
前項のGOTO，GOSUB命令では、プログラムの実行がその命令のところまでくると必ず指定した行番号に移りました（無条件ジャンプ）。

IF〈条件式〉THEN〈命令〉ELSE〈命令〉

真(成立する)のとき
偽(成立しない)のとき

IF～THEN～ELSE命令は、ある条件になったときに指定した行番号にジャンプする命令です。これを条件つきジャンプといいます。
IF～THEN～ELSE命令は、IFに続く条件が成立する（真）のときTHENに続く命令を実行し、成立しない（偽）のときはELSEに続く命令を実行します。

```
30 IF B=0 THEN GOTO 20
```

を追加してみましょう。これは変数Bが0のときは計算しないで、もう一度Bを入力し直す命令です。まず、実行（RUN）してみましょう。そして変数Bに0を入力してみてください。どうです？。
また変数Bの数をたずねてきましたね（20行へジャンプ）。今度は0でない数を入力してください。すると計算されて結果が表示されます（100行へジャンプ）。このようにIFに続く条件（条件式）によって、次に実行するプログラムを変えることができます。なお、ELSE文は、すぐ次の行を実行するとき、省略できます。

```
run
A=? 5
B=? 0
B=? 2
 7
 3
 10
 2.5
A=?
```

●条件式の種類

大小比較の条件式は、=，<，>の記号を使って右の表のように書きます。

複雑な条件式を作るときには、**アンド**（AND）や**オア**（OR）を使います。
たとえば、「AがBよりも大きくて、しかもCより小さい」という条件式は、

A>B AND A<C

と書くことができます。
また、「AがBよりも大きいか、Cよりも大きい」という条件式は、

A>B OR A>C

と書けます。

条件式の書きかた		
記号	意味	書きかた
=	AとBは等しい	A = B
>	AはBより大きい	A > B
<	AはBより小さい	A < B
<>または><	AとBは等しくない	A<>B
>=または=>	AはB以上か同じ	A>=B
<=または=<	AはB以下か同じ	A<=B

〈条件式1〉AND〈条件式2〉

〈条件式1〉と〈条件式2〉の両方が成立する

〈条件式1〉OR〈条件式2〉

〈条件式1〉と〈条件式2〉の一方が成立する

用語 **関係演算と論理演算子**
=，<，>のように数値の大小を比較する記号を“**関係演算子**”といいます。
また、AND，ORのように複数の条件式が成立しているかをチェックするものを、“**論理演算子**”といいます。

1・3・5 入力したキーのチェック……INKEY$（インキードル）

プログラムの実行中は、特別な命令の他にはキー入力を受けつけてくれません。
プログラムの実行中に、キー入力を受けつけてくれる命令には、**INPUT, INKEY$, STICK**などの命令があります。ここでは、**INKEY$**の使い方を覚えましょう。

```
A$ = INKEY$
```

押されたキーを覚えます。
押されたキーの文字を得ます。

INKEY$命令（正しくは関数）は、押したキーの文字を知るために使います。何もキーが押されていないときは" "（ヌルストリング：空（から）の文字）として判断されます。
行番号40～130の命令を追加と変更してください。正しく追加できましたか？では、実行（RUN）してみましょう。変数AとBに数を入力しても計算しませんね。
ここで「+」の記号をキー入力してみてください。どうです。たし算の計算結果が表示されましたね。
つまり、行番号40で押したキーの文字を変数A$に入れ、行番号50～80で何の記号かを判断しています。その結果、GOTO命令で指定の行番号の計算をします。このINKEY$は、ゲーム中にメニューを選んだりスタートボタンの代りに使ったりします。

```
10 INPUT "A=";A
20 INPUT "B=";B
30 IF B=0 THEN 20
40 A$=INKEY$
50 IF A$="+" THEN 100
60 IF A$="-" THEN 110
70 IF A$="*" THEN 120
80 IF A$="/" THEN 130
90 GOTO 40
100 PRINT A+B:GOTO 10
110 PRINT A-B:GOTO 10
120 PRINT A*B:GOTO 10
130 PRINT A/B:GOTO 10
140 GOTO 10
```

（40～90）命令を追加します
（100～130）:GOTO 10を追加します

ちょっと一言

■INKEY$関数

INKEY$関数はINPUT命令のようにキー入力があるまで待っていません。キーが押されていなくても次の命令を実行してゆきます。また、キー入力した文字は画面に表示されません。

■STICK命令

カーソルの押された方向を調べるだけならば、STICK関数があります。STICK関数については、第2章でも説明しています。
INKEY$……押されたキーの文字を得ます。（カーソルキーも調べることができます。）
STICK……カーソルキーの押された方向を調べます。

1・3・6　繰り返しに便利……FOR～NEXT

前ページのプログラムは、CTRL＋STOP と入力するか、エラーしない限りプログラムの実行が続きます。（これを無限ループといいます。）
これに対して、ある決められた回数だけ同じ命令を繰り返すものを有限ループといい、FOR～NEXT命令を使います。

```
10 N=1
20 PRINT N
30 N=N+1
40 IF N<=10 THEN GOTO 20
50 END
```

1から10までの数を画面に書くプログラムを作ってみましょう。

IF～THEN命令を使ってもいいですが、FOR～NEXT命令を使うとプログラムが簡単ですね。

```
10 FOR K=1 TO 10
20 PRINT K
30 NEXT K
40 END
```

FOR～NEXT命令は、FORとNEXTの間にある命令（ここでは行番号20命令）を、決められた回数だけくり返す命令です。

上記のプログラム例では変数Kは、1つづつ大きくなってゆきましたが、小さくすることもできます。

変数の変化する幅は、STEPという命令で指定できます。

STEPを省略すると、自動的に変化幅が1となります。

繰り返しのカウンタとして使う

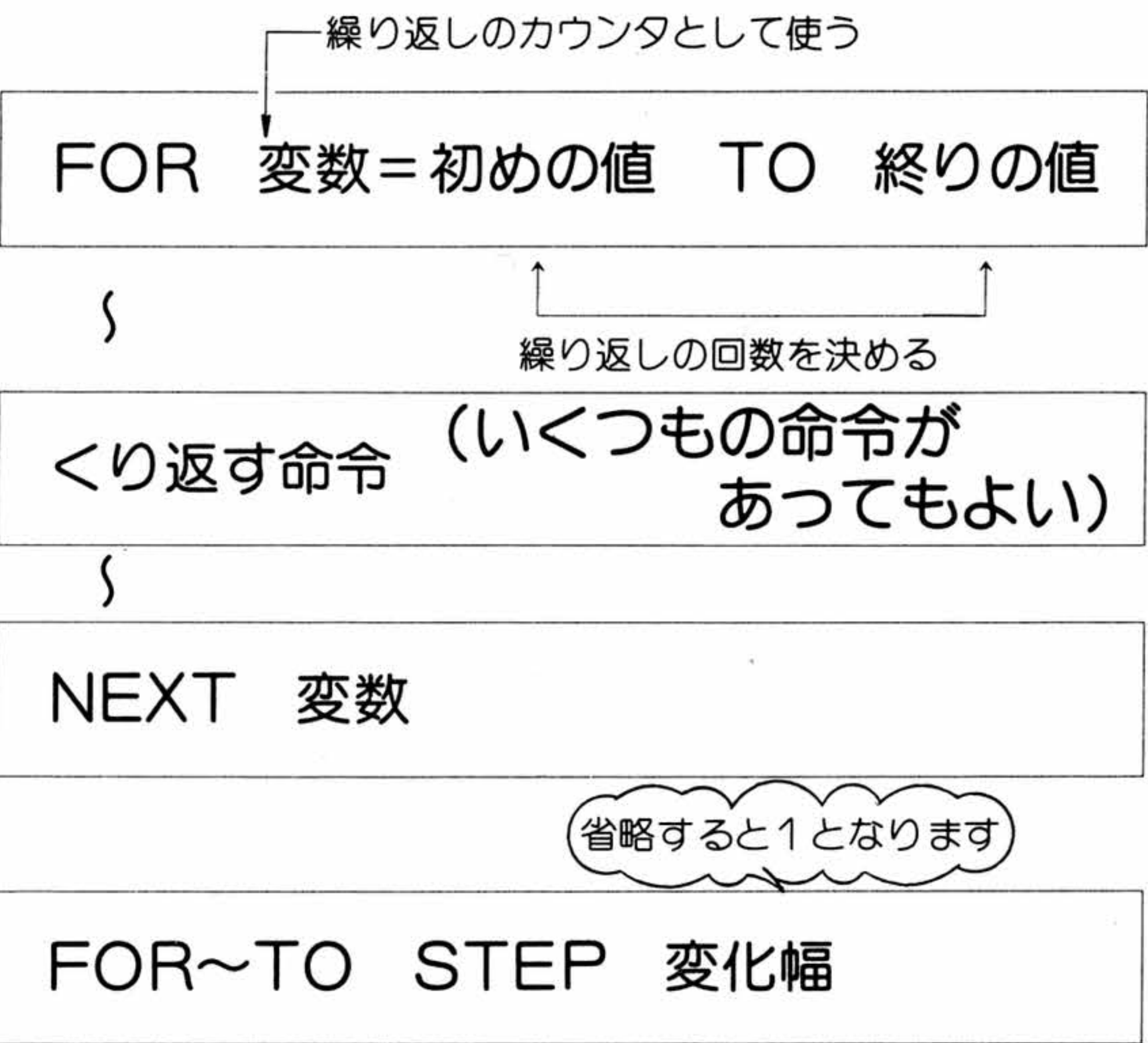
FOR　変数＝初めの値　TO　終りの値

繰り返しの回数を決める

くり返す命令　（いくつもの命令があってもよい）

NEXT　変数

省略すると1となります

FOR～TO　STEP　変化幅

ちょっと一言

■FOR～NEXTの組み合せ

FOR～NEXT命令は2重3重に重ねて使うことができます。しかし、FOR～NEXT命令が交差したり、FOR～NEXT命令の中に飛び込んではいけません。

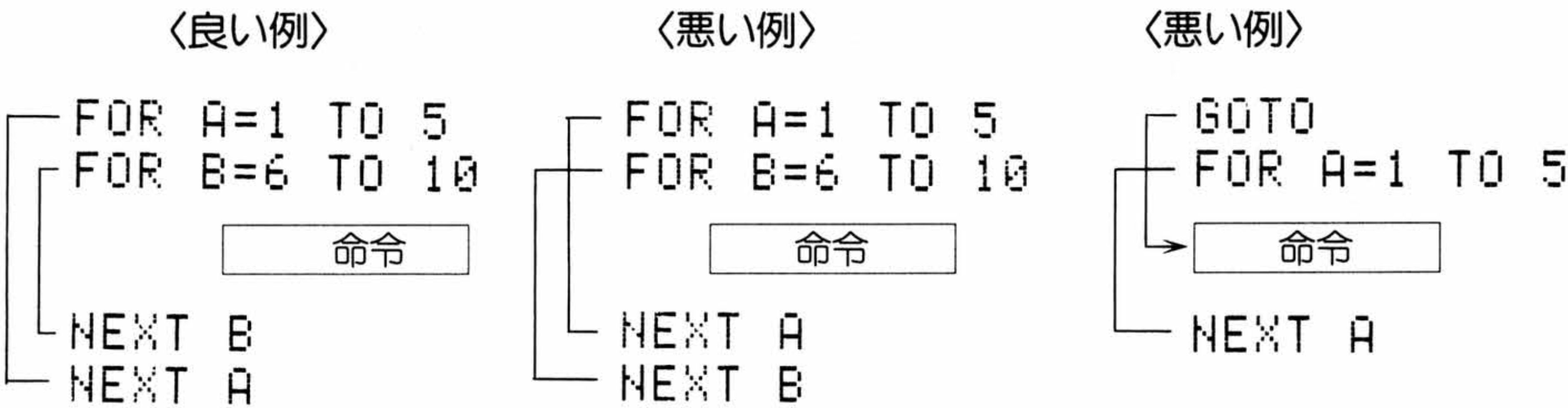
〈良い例〉

```
FOR A=1 TO 5
FOR B=6 TO 10
    命令
NEXT B
NEXT A
```

〈悪い例〉

```
FOR A=1 TO 5
FOR B=6 TO 10
    命令
NEXT A
NEXT B
```

〈悪い例〉

```
GOTO
FOR A=1 TO 5
    命令
NEXT A
```

1・3・7　多くの変数を使いたいとき……配列　DIM（ディメンジョン）

今までのプログラムではAとBという2つの変数で計算していましたが、もっと多くの数、たとえば20個も数があるときはプログラムが長くなってしまいます。
こんなときにはDIM命令を使うとプログラムを簡単にまとめることができます。

```
10 INPUT AA
20 INPUT AB
      .
      .
200 INPUT AT
      .
      .
```

20個の数を入力する

⇩

```
10 DIM A(19)
20 FOR N=0 TO 19
30 INPUT A(N)
40 NEXT N
      .
      .
```

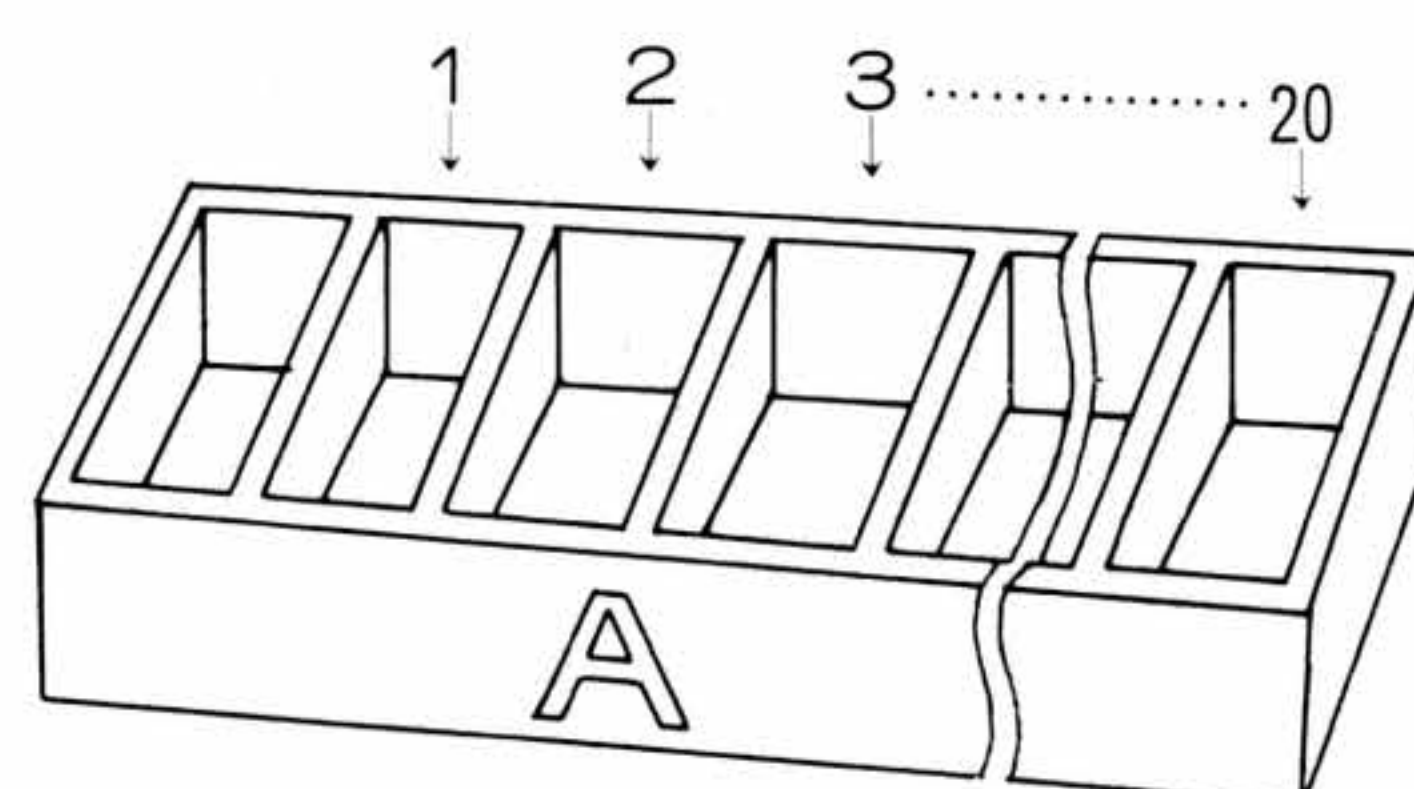
20個の数を入力する

たとえば、

DIM A (19)

を使うと、数の入力をするプログラムは右の図のようになります。
DIM命令の考え方は、右の図のようにマス目の箱を考えてください。DIM A (19) というのは、A（　）という変数に0～19の20個のマス目があり、最初に入力した数は変数A（0）に、次に入力した数は変数A（1）のマス目に順に入れられます。順序よく入れた数値はこのマス目の中に保管されて、ほかの命令で取り出すこともできます。たとえば、

（変数は、A(0)からA(19)まで使用できます。）

```
FOR N=0 TO 19
PRINT A (N)
NEXT N
```

とすれば、変数A（　）に入っている数が順に表示されます。
このように、DIM命令は変数が順序よくならべられて入っているので、「配列」と呼んでいます。

例．20個の数の平均を求めるプログラム

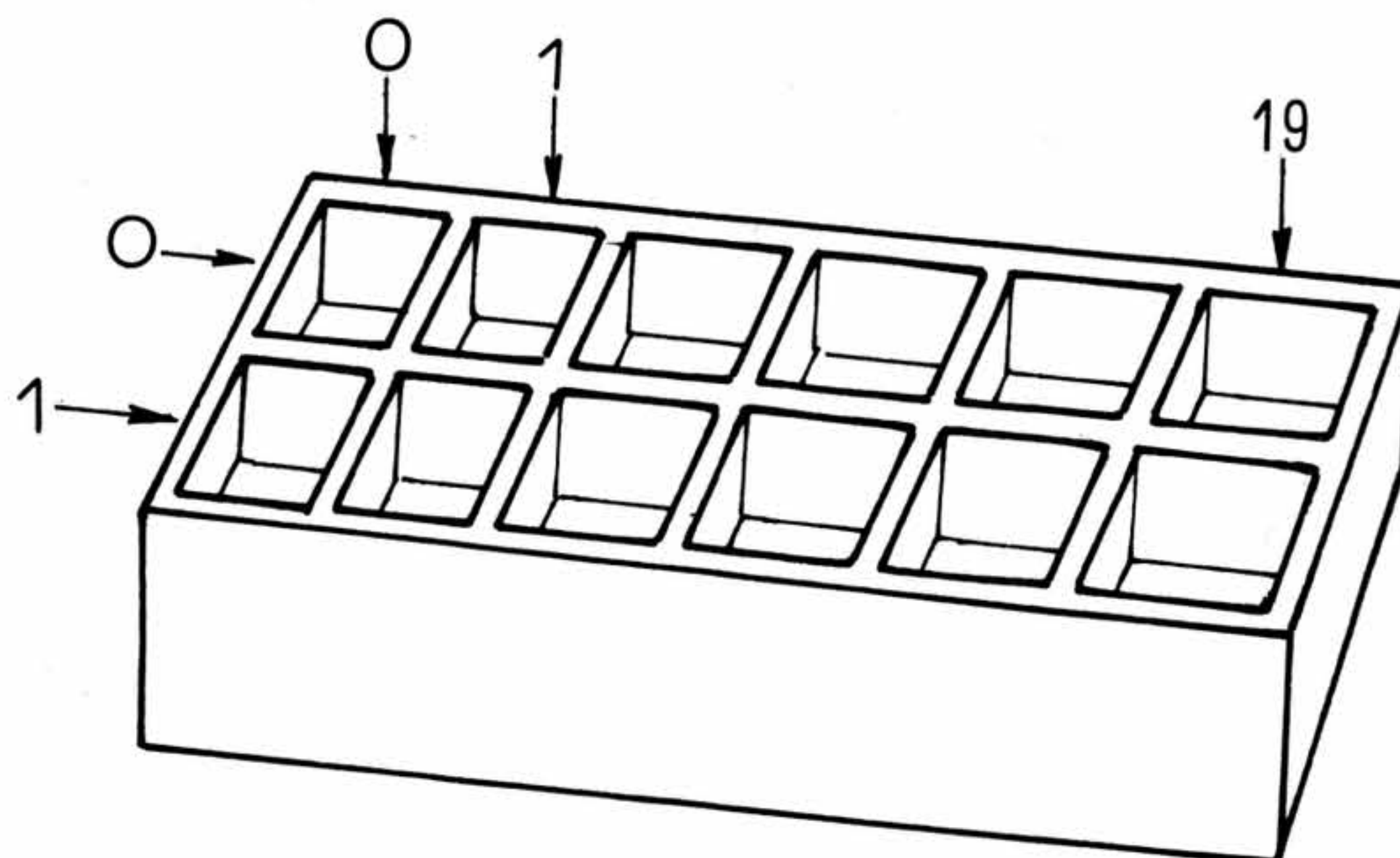
```
10 DIM A(19):B=0
20 FOR S=0 TO 19
30 PRINT S;"ﾊﾞﾝﾒ";
40 INPUT A(S)
50 B=B+A(S):NEXT S
60 B=B/20
70 PRINT "ﾍｲｷﾝﾊ";B;"ﾃﾞｽ"
80 END
```

●配列の次元

先ほど説明したDIM命令ではマス目が横（または縦）に順番に並んでいるもので、「一つの方向しかない」という意味で一次元の配列といいます。
これに対して、

DIM A (19, 1)

とすると、Aというマスは、横に0～19の20個のマスと、縦に0～1の2列のマス目が図のように並んだものとなります。これを二次元の配列といいます。
このDIM命令は、統計や家計簿、住所録に使うと便利です。

注意！　配列を使って計算ができますが、（　）の中の数字はDIM命令で使った数より大きくてはいけません。また、DIM命令は配列を使う前にします。

1．4　プログラムを保存するには……CSAVE，CLOAD

パソコンの電源を切ったり、リセットスイッチを押したら、せっかく入力したプログラムも消えてしまいますね。パソコンは、あなたが作ったプログラムをカセットレコーダに記録（保管）しておき、必要な時に読み込むことができます。

●記録の方法と読み込み方

カセットレコーダの使い方、およびCSAVE，CLOAD命令の使い方については取扱説明書で詳しく説明しています。ここでは、プログラムの記録（セーブ）と読み込み（ロード）の方法には、どんな種類があるのかを表で説明します。

命令の詳しい説明は第3章をご覧ください。また、ファイルは、1．5．7で説明します。

命令	CSAVE CLOAD／CLOAD？	SAVE LOAD	BSAVE BLOAD
目的・特長	• ベーシックで作ったプログラムを短かくして、カセットレコーダに記録する。 (圧縮コードを使うといいます。) • 記録／読み込みが簡単にできます。 • カセットレコーダ用の命令です。	• ベーシックで作ったプログラムをそのままカセットレコーダに記録する。 • プログラムの混合もできます。 (アスキーセーブといいます。) • フロッピーディスクにも使える命令です。	• 機械語で作ったプログラムを記録します。 • フロッピーディスクにも使える命令です。
記録方法	カセットを録音状態にする ↓ CSAVE"ファイル名" リターン ↓ プログラムの記録 ↓ カセットを再生状態にする ↓ CLOAD？ リターン ↓ 正確に記録されたか確認	カセットを録音状態にする ↓ SAVE"ファイル名" リターン ↓ プログラムの記録	カセットを録音状態にする ↓ BSAVE"ファイル名"、開始番地、終了番地 リターン ↓ プログラムの記録
読込み方法	カセットを再生状態にする ↓ CLOAD"ファイル名" リターン ↓ プログラムの読み込み	カセットを再生状態にする ↓ LOAD"ファイル名" リターン ↓ プログラムの読み込み ※カセットレコーダを使用した場合、アスキー形式で記録されます。MERGE命令で読み込むと、プログラムの混合ができます。	カセットを再生状態にする ↓ BLOAD"ファイル名" リターン ↓ 同じプログラムの読み込み

1．5　こんなことも　できるよ

前項までは、主にBASIC言語を使ってプログラムの作り方を説明してきました。この項では、MSX BASIC特有の命令を使ったプログラムの作り方を説明します。MSX BASICは、スプライト機能や音楽機能を備えていますので、楽しいプログラムを作ってください。

1．5．1　楽しい演奏……PLAY（プレイ）

MSX BASICでは、３重和音の音楽の演奏ができます。この演奏を行うのがPLAY命令です。

●音階を出す。

PLAY "CDEFGAB" [リターン]

と打ち込んでください。

ドレミファソラシと演奏されますね。このPLAYの後ろに演奏したい曲の音階を表わす文字や記号を入れると、曲が演奏できます。

PLAY "曲を表わす文字・記号"

演奏しなさい！　演奏する曲

●和音を演奏する

PLAY "C", "E", "G" [リターン]

と打ち込んでください。ドミソの和音が演奏されましたね。このように３つまでの音を同時に出して、和音にもできます。

和音の区切りは、カンマ（，）で行ないます。

PLAY "曲", "曲", "曲"

３重和音の演奏

PLAY命令の詳しい説明、およびサンプルプログラムについては、第２章、第３章をご覧ください。

1．5．2　ゲームの音を作ろう……SOUND

PLAY命令で曲の演奏をしましたが、もっと細かな音の変化を出したり、ゲームなどの効果音を作るときは、**SOUND**命令を使います。

SOUND命令は、パソコン内の音を作り出すSOUND ICに、出力したい音のデーターを入力することにより、さまざまの音（波の音、効果音など）を発生させることができます。このICでは、次の3つの機能を制御できます。

●連続した単音の発生

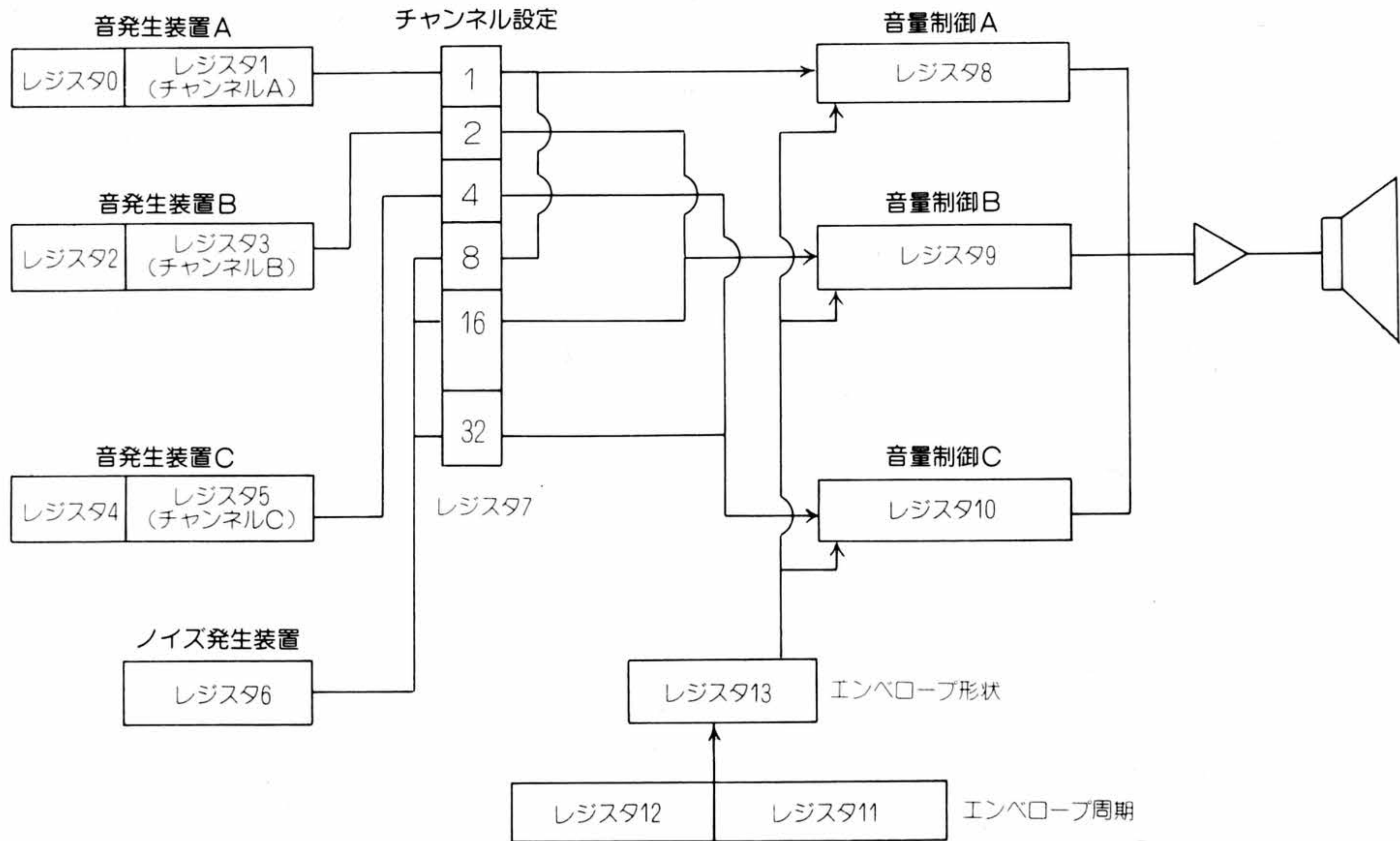
●ノイズ（波の音など）の発生

●音量を自動的に変化させる

音を出す　メモリの番号　音の情報

SOUND〈レジスタ番号〉,データ

SOUND命令の詳しい使い方、サンプルプログラムは、第3章、第4章をご覧ください。

音発生装置A　レジスタ0　レジスタ1（チャンネルA）
音発生装置B　レジスタ2　レジスタ3（チャンネルB）
音発生装置C　レジスタ4　レジスタ5（チャンネルC）
ノイズ発生装置　レジスタ6
チャンネル設定　1　2　4　8　16　32　レジスタ7
音量制御A　レジスタ8
音量制御B　レジスタ9
音量制御C　レジスタ10
レジスタ13　エンベロープ形状
レジスタ12　レジスタ11　エンベロープ周期

〈SOUND IC内のレジスタの構成は次のようになっています。〉

<table>
<tr><th colspan="2">レジスタ</th><th colspan="24">デ　ー　タ</th></tr>
<tr><td>R0</td><td rowspan="2">チャンネルA周波数</td><td colspan="24">0～15</td></tr>
<tr><td>R1</td><td colspan="24">0～255</td></tr>
<tr><td>R2</td><td rowspan="2">チャンネルB周波数</td><td colspan="24">0～15</td></tr>
<tr><td>R3</td><td colspan="24">0～255</td></tr>
<tr><td>R4</td><td rowspan="2">チャンネルC周波数</td><td colspan="24">0～15</td></tr>
<tr><td>R5</td><td colspan="24">0～255</td></tr>
<tr><td>R6</td><td>ノイズ周波数</td><td colspan="24">0～63</td></tr>
<tr><td rowspan="2">R7</td><td rowspan="2">チャンネル設定</td><td colspan="12">ノ　イ　ズ</td><td colspan="12">チャンネル</td></tr>
<tr><td colspan="4">C</td><td colspan="4">B</td><td colspan="4">A</td><td colspan="4">C</td><td colspan="4">B</td><td colspan="4">A</td></tr>
<tr><td>R8</td><td>チャンネルA音量</td><td colspan="24">音量（0～15、16でエンベロープ）</td></tr>
<tr><td>R9</td><td>チャンネルB音量</td><td colspan="24">音量（0～15、16でエンベロープ）</td></tr>
<tr><td>R10</td><td>チャンネルC音量</td><td colspan="24">音量（0～15、16でエンベロープ）</td></tr>
<tr><td>R11</td><td rowspan="2">エンベロープ周期</td><td colspan="24">0～255</td></tr>
<tr><td>R12</td><td colspan="24">0～255</td></tr>
<tr><td>R13</td><td>エンベロープ形状</td><td colspan="12"></td><td colspan="3">E3</td><td colspan="3">E2</td><td colspan="3">E1</td><td colspan="3">E0</td></tr>
</table>

※一度、レジスタに値を設定すると、新しいデータを設定するまでその値は変わりません。

1.5.3　画面を切替える……SCREEN、画面モード

MSX BASICでは、画面の状態（モード）を切替えて、プログラムを表示したり、絵を絵がいたりできます。

この画面のモードを切替える命令をSCREEN（スクリーン）命令といいます。なお、今までプログラムを作っていた画面は **〝テキストモード〟** といいます。

SCREEN 〈画面のモード〉

①テキストモード（SCREEN 0、SCREEN 1）

画面にプログラムや文字を表示する画面です。なお、グラフィック命令は使えません。

- **SCREEN 0……40×24テキストモード**
  画面に表示できる文字数は、40文字(最大)×24行です。1行に表示できる文字数は、WIDTH命令で変えることができます。（ MSX2 では、80文字×24行も指定できます）
  なお、1文字は、6ドット×8ドットで構成されていますので、文字によっては表示が欠けることもあります。(例〝月″など)
- **SCREEN 1……32×24テキストモード**
  画面に表示できる文字数は、32文字(最大)×24行です。1行に表示できる文字数は、WIDTH命令で変えることができます。なお、1文字は8ドット×8ドットで構成されています。

②グラフィックモード（SCREEN 2～SCREEN 8）

画面に線や円などの図を描くことができます。プログラム実行中（プログラムモード）でしか表示できません。また、INPUT命令を実行したときや、エラーが発生したときはテキストモードに戻ります。（SCREEN 4～8は、MSX2 でのみ使用でき、SCREEN 7と8はVRAMの容量が128KBの機種のみです）

- **SCREEN 2……高解像度グラフィックモード**
  256×192ドットで画面を構成します。
- **SCREEN 3……マルチカラーモード**
  64×48ブロック（4×4ドットを1ブロックとする）で画面を構成します。
- **SCREEN 4……高解像度グラフィックモード**
  256×192ドットで画面を構成します。SCREEN 2のスプライト機能を拡張しています。

注意：SCREEN 2と4では画面を（0，0）から右へ8画素ごとに分割して表示色を管理しています。分割された8画素ごとに、背景色を含めて2色まで使うことができます。3色目を指定すると、先に指定したところの色が変化することがあります。

●SCREEN 5……ビットマップグラフィックモード
256×212ドットで画面を構成します。1点ごとに色を設定できます（512色中16色使用可）。

●SCREEN 6……ビットマップグラフィックモード
512×212ドットで画面を構成します。1点ごとに色を設定できます（512色中4色使用可）。

●SCREEN 7……ビットマップグラフィックモード
512×212ドットで画面を構成します。1点ごとに色を設定できます（512色中16色使用可）。

●SCREEN 8……ビットマップグラフィックモード
256×212ドットで画面を構成します。1点ごとに色を設定できます（256色使用可）。

■テキストモード

SCREEN 0

SCREEN 1

※ MSX の機種は、電源を"入"にしたときはSCREEN 1に設定されています。

■グラフィックモード

SCREEN 2
SCREEN 3
SCREEN 4

※SCREEN 3は、64×48ブロックで構成されていますが、グラフィック命令の座標指定は、255×191で行ないます。

SCREEN 5
SCREEN 8

※SCREEN 5では512色中16色、SCREEN 8では256の色が使えます。

SCREEN 6
SCREEN 7

※SCREEN 6では512色中4色、SCREEN 7では512色中16色が使えます。

1．5．4　画面の色を変えよう………COLOR(カラー)

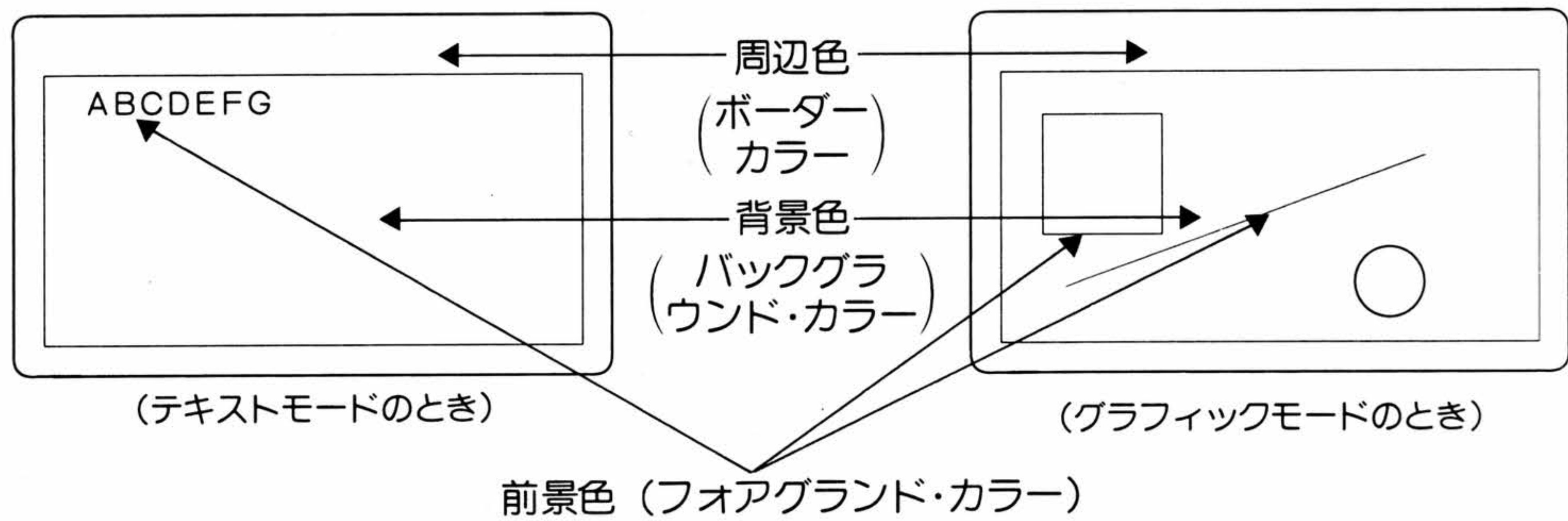
COLOR　15，1，4　[リターン]
と打ち込んでみてください。
画面の色が変わりましたね。この、画面の色を変える命令をCOLOR(カラー)命令といいます。
そしてCOLORの後に続く数字（カラーコードといいます）によって画面の色を決めます。

色はカラーコードで指定します。

COLOR （前景色のカラーコード），（背景色のカラーコード），（周辺色のカラーコード）

カラーコードと色の対応は、次のとおりです。

カラーコード	色	カラーコード	色	カラーコード	色	カラーコード	色
0	透明(周辺色)	4	暗い青	8	赤	12	暗い緑
1	黒	5	明るい青	9	明るい赤	13	紫
2	緑	6	暗い赤	10	黄色	14	灰
3	明るい緑	7	水色	11	明るい黄	15	白

（テキストモードのとき）　　（グラフィックモードのとき）

いろいろな色を指定してみてください。

注意！ MSX2では、カラーコードと画面に表示される色の対応を変えることができます。詳しくは第3章のColor命令の項を参照してください。

ちょっと一言

キーボードのファンクションキー[F1]、[F6]には次の命令が入っています。

[F1]………COLOR
[F6]………COLOR 15，4，7 [RETURN]

1．5．5　いろんな図を書こう……LINE，PSET，CIRCLE，PAINT

画面をグラフィックモードに切替えて、絵を書いてみましょう。

● 点を描く　（PSET）

PSET命令は、グラフィック画面に点を描きます。

右のプログラムは、画面に適当に点を描きます。

（止めるときは CTRL + STOP キーを押してください。）

SCREEN 3では、4×4ドットの1ブロック内なら、どの座標を指定しても同じ位置に点が打たれます。

（NEW命令を実行してから入力してください。）

点の位置　点の色

PSET（〈X座標〉，〈Y座標〉），〈カラーコード〉

```
10 SCREEN 2
20 X=RND(1)*255
30 Y=RND(1)*191
40 C=RND(1)*15
50 PSET(X,Y),C
60 GOTO 20
```

● 線を描く……LINE

線は点の集まりですから、FOR〜NEXT文とPSET命令を使えば線を描くことができます。

だけどLINE命令を使うと、簡単に線を描くことができます。

（NEW命令を実行してから入力してください。）

マイナス記号です

LINE（開始点）ー（終了点），〈カラーコード〉，（B BF）

線の端と端を指定します。　線の色　四角を描くときに指定します。

```
10 COLOR ,1,1
20 SCREEN 2
30 C1=RND(1)*15
40 C2=RND(1)*15
50 FOR X=0 TO 255 STEP 8
60 LINE(X,0)-(255-X,191),C1
70 LINE(X+4,0)-(251-X,191),C2
80 NEXT X
90 FOR Y=0 TO 191 STEP 8
100 LINE(0,191-Y)-(255,Y),C1
110 LINE(0,187-Y)-(255,Y+4),C2
120 NEXT Y
130 GOTO 30
```

●円を描く……CIRCLE（サークル）

CIRCLE命令を使うと円を描くこともできます。
ただし、円を描く命令は次のように指定し、角度の単位はラジアンです。

360°（度）＝2π

＝2×3.141659……（ラジアン）

〈円を連続して描きます〉

```
10 COLOR 15,1,1
20 SCREEN 2:C=2
30 FOR X=0 TO 255 STEP 16
40 Y1=SIN(4*3.1416*X/255)*30+50
50 Y2=COS(4*3.1416*X/255)*30+140
60 CIRCLE(X,Y1),7,C
70 CIRCLE(X,Y2),7,C+1
80 C=C+1:IF C>14 THEN C=2
90 NEXT X
100 GOTO 100
```

CIRCLE（中心の座標），円の半径，円の色，開始角，終了角，比率

円の位置

円の大きさ

円弧を指定するとき

楕円を描くとき

●色を塗る……PAINT（ペイント）

LINE 命令や CIRCLE 命令を使った図形を、1つの色で塗りつぶすとき、PAINT命令を使います。
ただし、色の塗りつぶしは囲まれた図形でないと画面全体を塗ることになってしまいます。
また、SCREEN2のときは、塗りつぶす色と境界色は同じ色でなければなりません。

〈円を塗りつぶします〉

```
65 PAINT(X,Y1),C
75 PAINT(X,Y2),C+1
```

PAINT（塗る開始点の座標），〈指定色〉，〈境界色〉

囲まれた図形の中の位置

塗りつぶす色

図形の色

1．5．6　絵を動かそう（スプライト）……SPRITE（スプライト）機能

画面に描いた絵を動かすには、描いた図の座標を変えながら１個の点ごとにPSET、PRESET　命令を繰り返すこともできますが、個々の点を動かしていたのではプログラムの実行が大変です。
このようなとき、MSX BASICのSPRITE機能を使います。SPRITE機能は、絵を1つのパターン（縦8×横8または縦16×横16のドットを組み合わせた絵）として扱い、パターン全体を動かすことができる機能です。

SPRITE機能はパターンの形を決めるSPRITE$命令と、このパターンを表示する位置を決めるPUT SPRITE命令からなります。（MSX2では他に、COLOR SPRITE命令も使えます。）

MSX BASICでは、0〜31の仮想画面があり、それらをプレーンといいます。そして、１つのプレーンには1つのスプライトを表示できます。

通常、書き込む画面
プレーン31
プレーン1
プレーン0

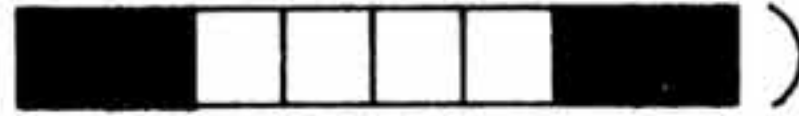
①画面モードの設定（SCREEN）

パターンの大きさは、２種類あります。これを指定するのがSCREEN命令で８×８または16×16ドットで使ったパターンを「標準」「拡大」で指定できます。（拡大は縦横各２倍となります。）
なお、スプライト機能はSCREEN 0では使用できません。

例　SCREEN　2，1

画面のモードは
1〜3

SCREEN〈画面のモード〉、〈スプライトサイズ〉

番号	スプライトサイズ	
0	8×8	標準
1	8×8	拡大
2	16×16	標準
3	16×16	拡大

②パターンの設定（SPRITE$）

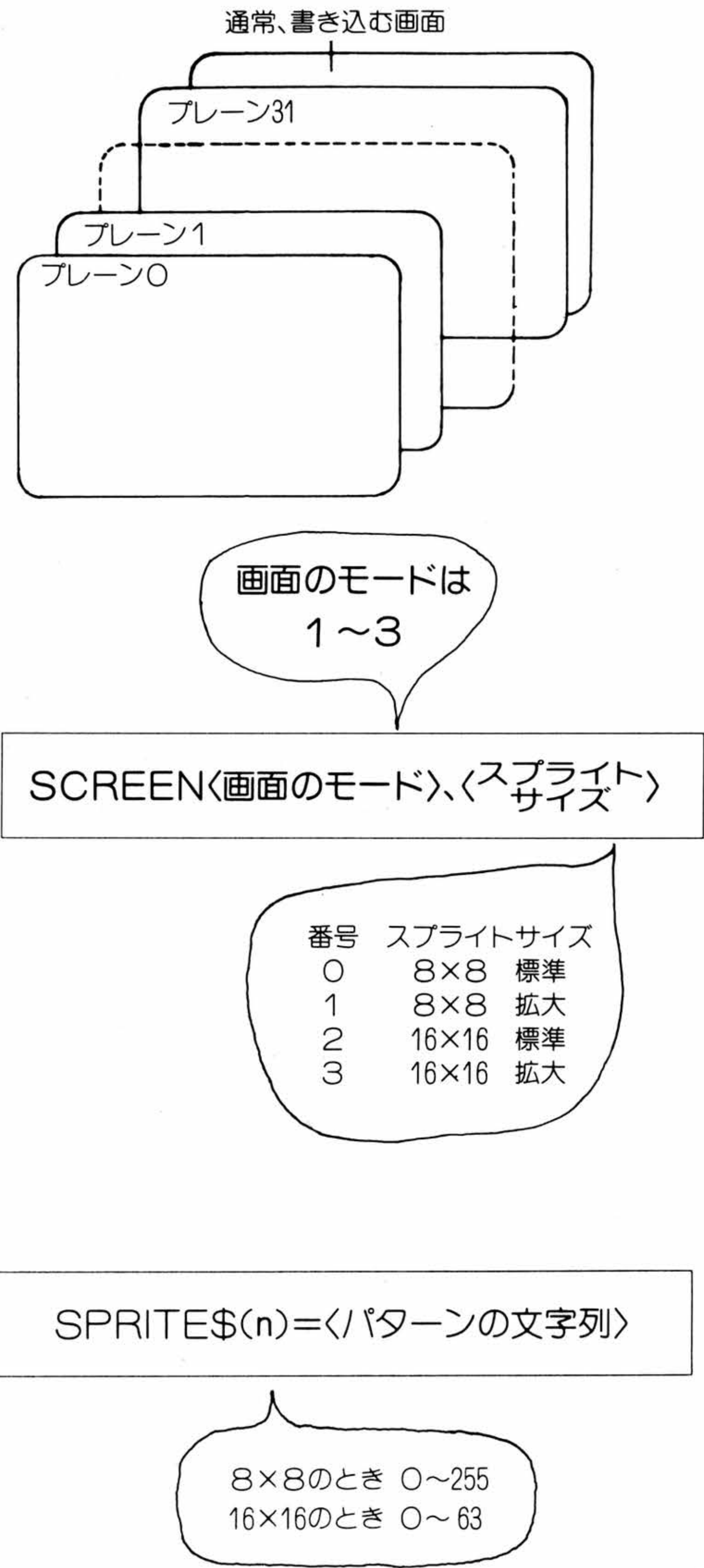
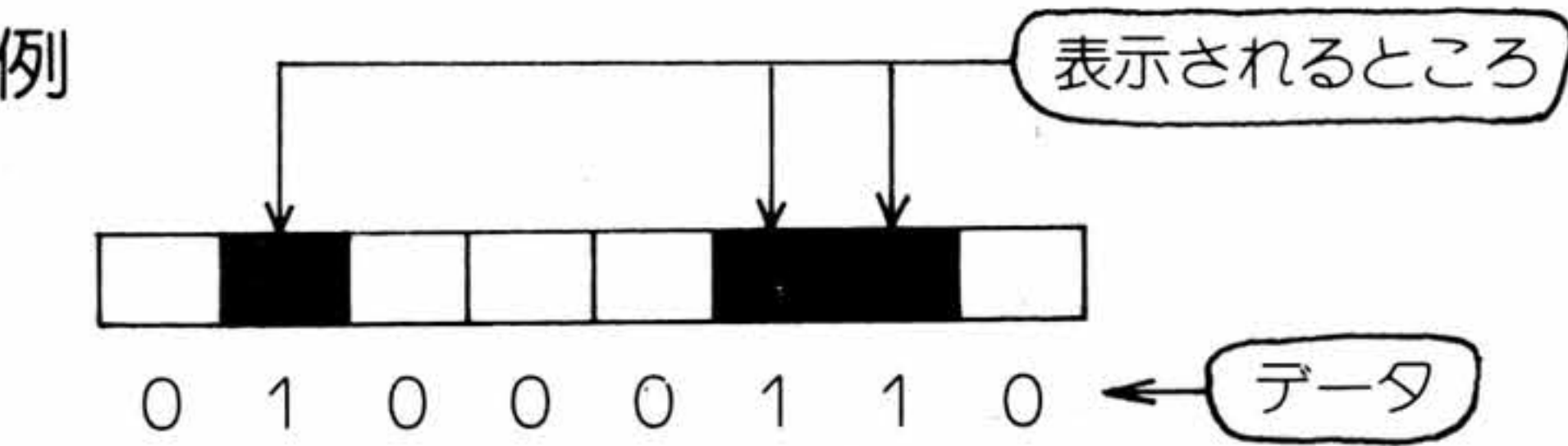
パターンは８×８または16×16ドットで作ります。このパターンの各ドットについて「表示するのか（データ１）、表示しないのか（データ０）を、文字列にして指定します。パターンは一番上の列から下へ順に８つのデータを加えて指定します。

SPRITE$(n)=〈パターンの文字列〉

8×8のとき　0〜255
16×16のとき　0〜63

例

表示されるところ

0　1　0　0　0　1　1　0　←データ

用　語　スプライト：SPRITE（妖精（ようせい））

プレーン：PLANE（平面、面）

右のパターンを例にして説明します。

1番上のパターン　（■□□□□□■■ の並び：■■□□□□■■）　を表わすには、左から

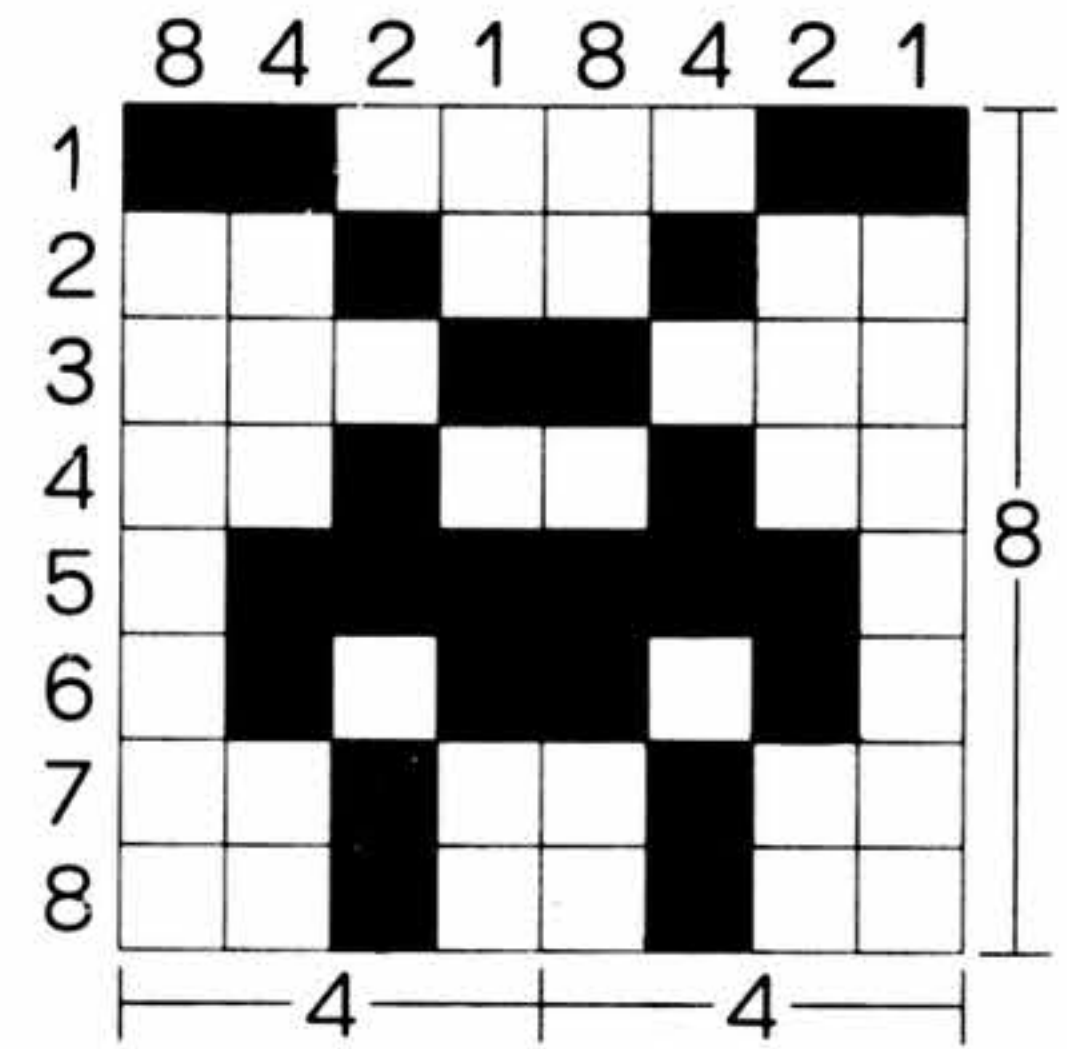
　11000011

というデータが使われます。これをパターンデータとして使うには、CHR$(&B11000011)とします。2番目以降も同じようにして、各データを+でつなぎます。

そして、パターンを設定するにはSPRITE$( )命令を使います。右のパターンは、

SPRITE(n)=CHR$(&B11000011)+CHR$(&B00100100+CHR$(&B00011000)+CHR$(&B00100100)+CHR$(&B01111110)+CHR$(&B01011010)+CHR$(&B00100100)+CHR$(&B00100100)

で設定されます。この設定のしかたは、2進数を使ったものです。2進数（&B……）の代りに、8進数（&O…）、16進数（&H……）に変えても設定できます。

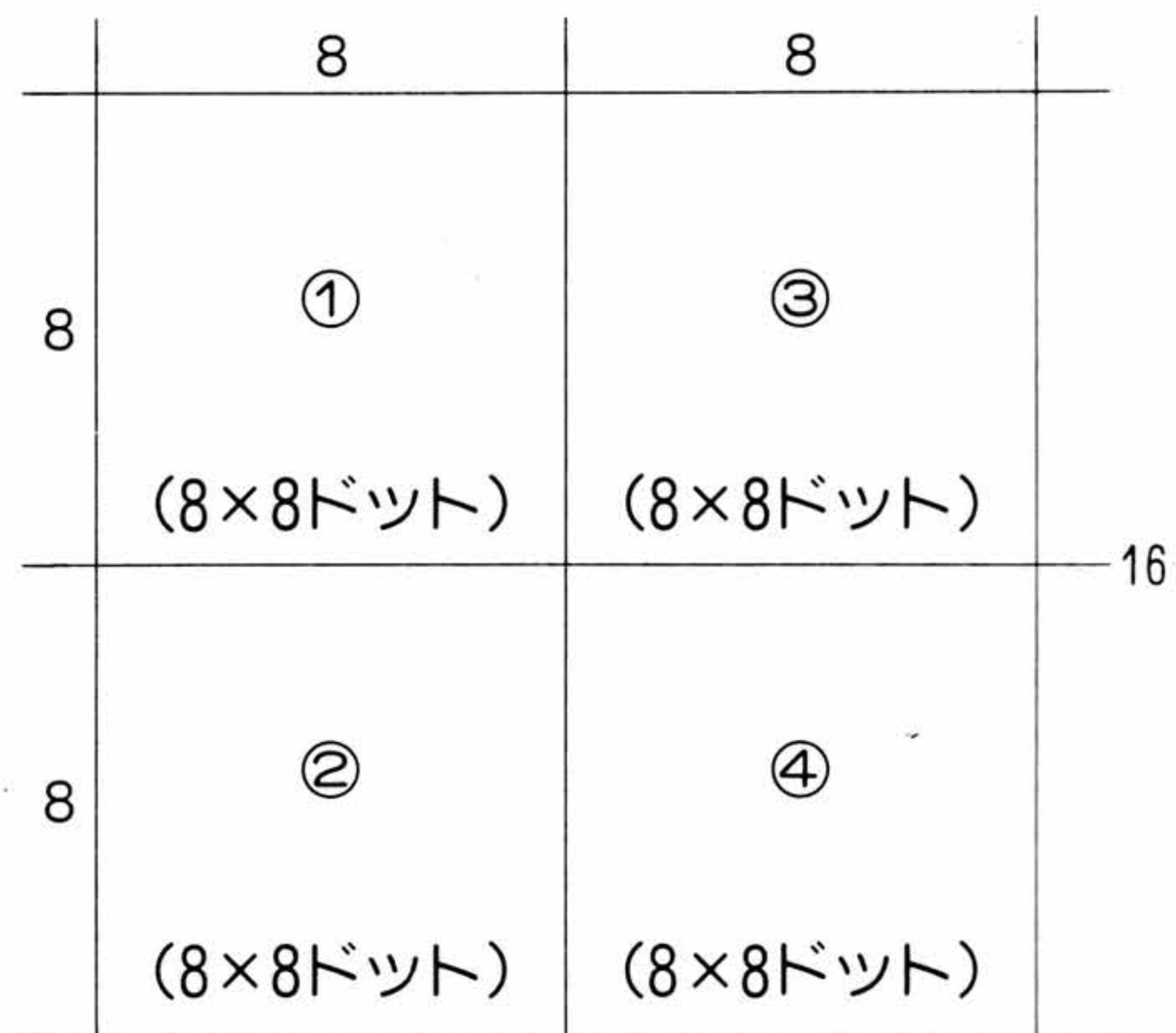
	8	4	2	1	8	4	2	1
1	■	■	□	□	□	□	■	■
2	□	□	■	□	□	■	□	□
3	□	□	□	■	■	□	□	□
4	□	□	■	□	□	■	□	□
5	□	■	■	■	■	■	■	□
6	□	■	□	■	■	□	■	□
7	□	□	■	□	□	■	□	□
8	□	□	■	□	□	■	□	□

16×16のスプライトパターンの場合は、4つの8×8のスプライトパターンを次の順に連結して設定すれば良いのです。

SPRITE$(n)=〈①の設定〉+〈②の設定〉+〈③の設定〉+〈④の設定〉

（①～④は右の図の番号）

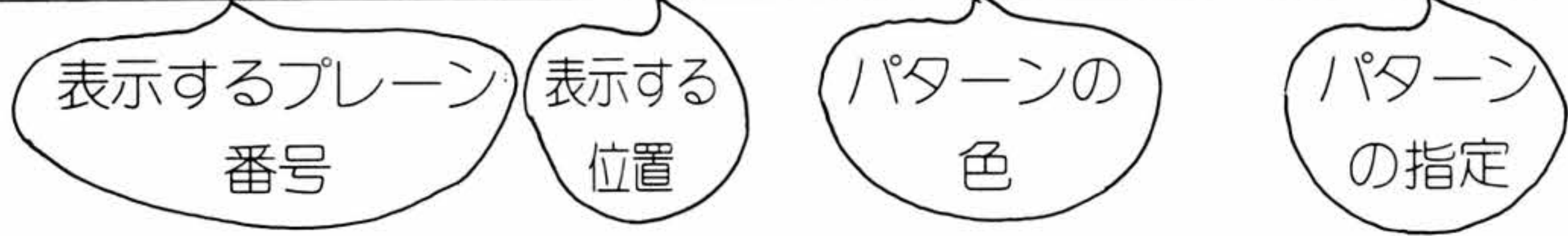
パターンは8×8の時255個、16×16の時63個まで指定できるので、どのパターンかわかるようにSPRITE$(n)のnの番号で区別します。

	8	8
8	① (8×8ドット)	③ (8×8ドット)
8	② (8×8ドット)	④ (8×8ドット)

（全体16）

③パターンの表示（PUT SPRITE）

設定したパターンをプレーンに書き込むにはPUT SPRITE命令を使います。

PUT SPRITE 〈プレーン番号〉,〈X，Y〉,〈カラーコード〉,〈パターン番号〉

- 〈プレーン番号〉：表示するプレーン番号
- 〈X，Y〉：表示する位置
- 〈カラーコード〉：パターンの色
- 〈パターン番号〉：パターンの指定

例　PUT SPRITE　1,(128,100),8,1

ちょっと一言

スプライトパターンは1プレーンに1つしか書けません。

多くのプレーンを使ってスプライトパターンを表示するとき、画面の横方向（Y座標がほぼ同じとき）でパターンが重なると、5つ以上のパターンは表示されません。

また、表示するプレーンには優先順位があるので、重なった時、プレーンの番号の小さいものが優先して表示されます。（MSX2では、8つまで表示することができます。）

1．5．7　ファイルってなに？

ファイルとは、意味を持つ情報の集まりです。カセットテープにセーブ／ロードしたプログラムやデータは、ファイルとして扱われます。

パソコン
データの入出力
カセットレコーダ
プリンタ
ファイル
バッファを使います。

①ファイル番号

プログラム中でデータの入出力を行うときは、あらかじめOPEN文を使ってファイルの使用開始を宣言し、ファイルの番号と入出力のときに使うメモリの領域（バッファといいます）を確保します。

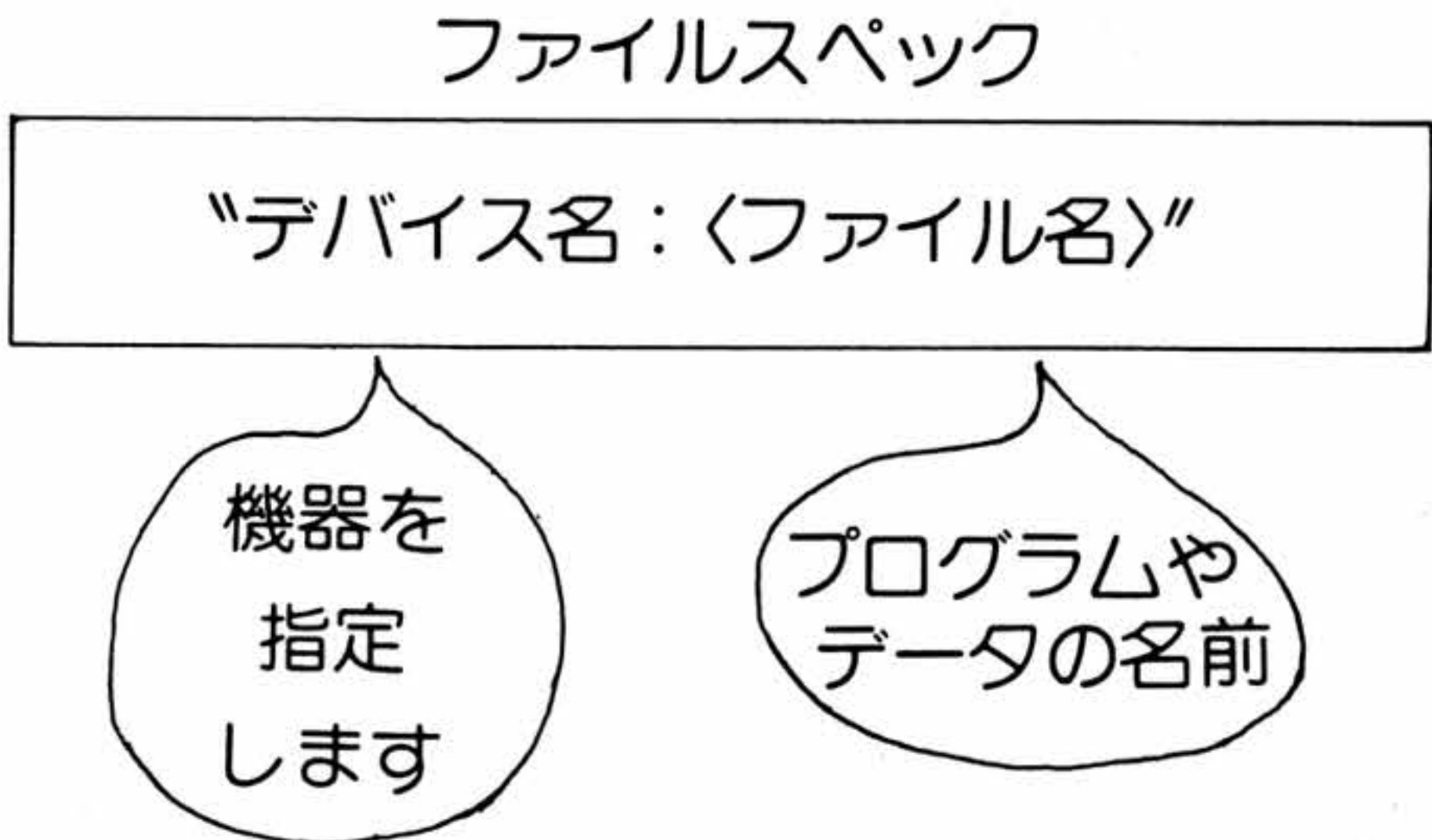
②ファイルスペックって？

ファイルスペックは、ファイルの入出力を行なう機器や、ファイルするデータやプログラムの名前を指定します。

ファイルスペックは、必ず引用符で囲み（" "）〈デバイス名〉、〈ファイル名〉は省略できる場合もあります。

ファイルスペック

"デバイス名：〈ファイル名〉"

機器を指定します
プログラムやデータの名前

●デバイス名

〈デバイス名〉は入出力機器を表すもので次のようなものがあります。

デバイス名	入出力デバイス	使用できるモード	
		INPUT	OUTPUT
CAS：	カセットレコーダ	○	○
CRT：	テキスト画面	×	○
GRP：	グラフィック画面	×	○
LPT：	プリンタ	×	○
	フロッピーディスク	○	○
MEM：	RAMディスク	○	○

※MEM：は MSX2のみ。フロッピーディスクには「A:」「B:」を使います（最高A～Hまで）。

※本書の第3章では、MSX BASICの命令のほかに、参考としてDisk BASICの命令を記載しています。データ、プログラムを入出力するときの参考としてください。

●ファイル名

〈ファイル名〉とはプログラムファイルやデータファイルに付ける名前です。ファイル名を必要とするのは、カセットレコーダとフロッピーディスクに入出力を行う場合で、その他の場合は省略することができます。カセットレコーダの〈ファイル名〉は6文字です。〈ファイル名〉が6文字より小さい場合は空いた部分にスペースが入ります。また〈ファイル名〉が6文字を越えて指定された場合は、先頭から6文字が〈ファイル名〉と見なされ、それ以降の文字は無視されます。

フロッピーディスクとRAMディスク（MSX2のみ）の〈ファイル名〉は8文字まで指定することができます。また〈ファイル名〉の後に、拡張子として3文字までの文字が指定できます。

1．5．8 割込みについて

●割込みとは？

GOTO，GOSUB文は決った手順で指定の行番号へジャンプしますが、プログラムの実行中に、何らかの条件がそろったときに、プログラムの実行を中断して別のプログラムを実行させることもできます。これを割込みといいます。

〈メインルーチン〉 〈割込み処理ルーチン〉

RETURN

たとえば、プログラム実行中に1秒ごとに時間表示させるときには、メインルーチンを実行しているときに、1秒ごとに割込みをしてプログラムの実行を止め、割込み処理ルーチンで時間表示をさせます。そして時間表示を行った後、もとのメインルーチンの実行を再開します。

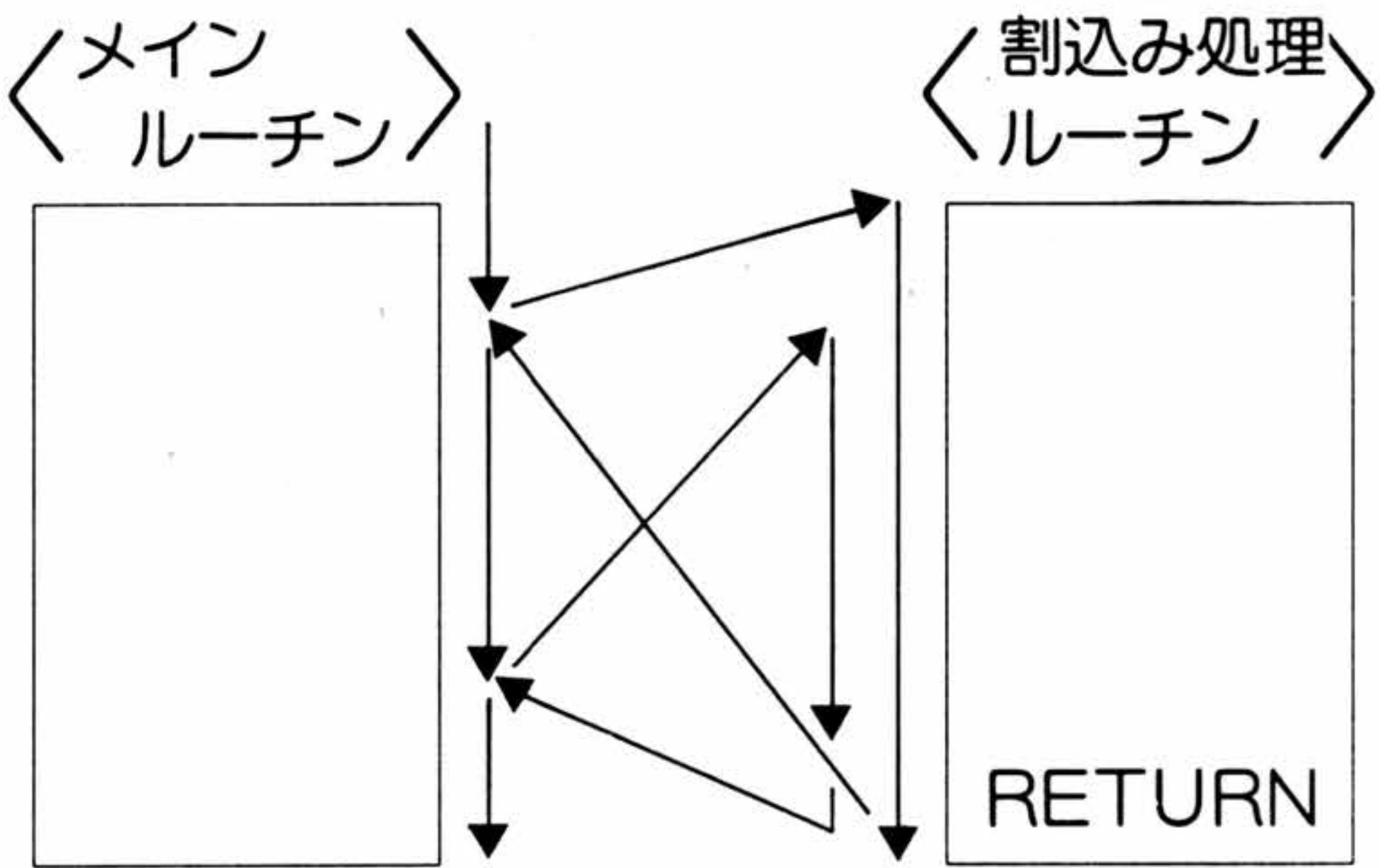
割込み処理とGOSUB文によるサブルーチンとの違いは次のとおりです。

GOSUB…… ●決まった手順でサブルーチンへジャンプします。
　　　　　 ●ジャンプする条件は、指定しなくてもよい。

割 込 み…… ●ある特定の条件がそろったときにサブルーチン（割込み処理ルーチン）へジャンプしますので、手順を決めることはできない。
　　　　　 ●ジャンプする条件とその処理ルーチンをあらかじめ設定する必要がある。

●割込みの種類

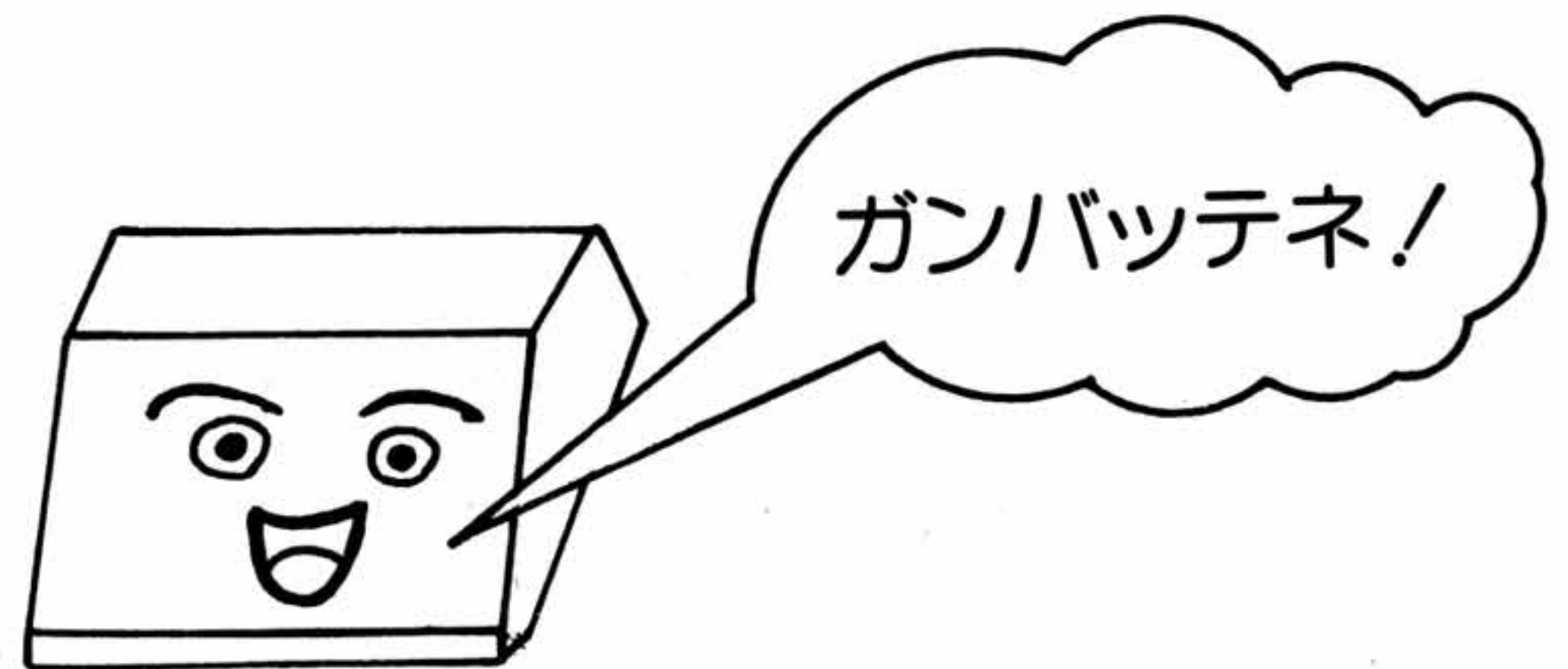
MSX BASICが用意している割込みは次のとおりです。また割込みの優先順序も次のとおりです。各々の命令については第2章、第3章をご覧ください。

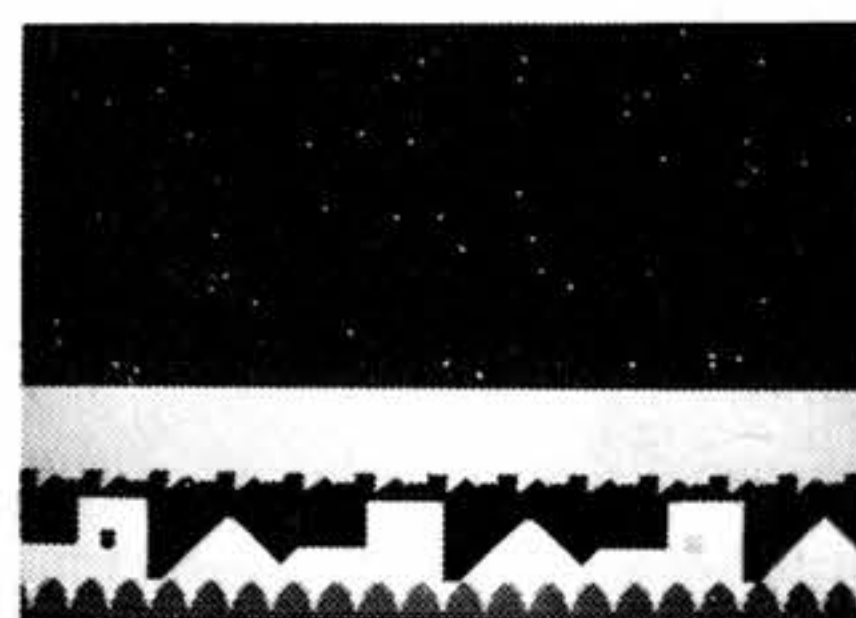
ファンクションキー割込み　（ON KEY GOSUB）
ストップキー割込み　（ON STOP GOSUB）
スプライトの衝突割込み　（ON SPRITE GOSUB）
ジョイスティックのトリガ割込み　（ON STRIG GOSUB）
インターバルタイマ割込み　（ON INTERVAL GOSUB）

第2章
プログラムの作り方

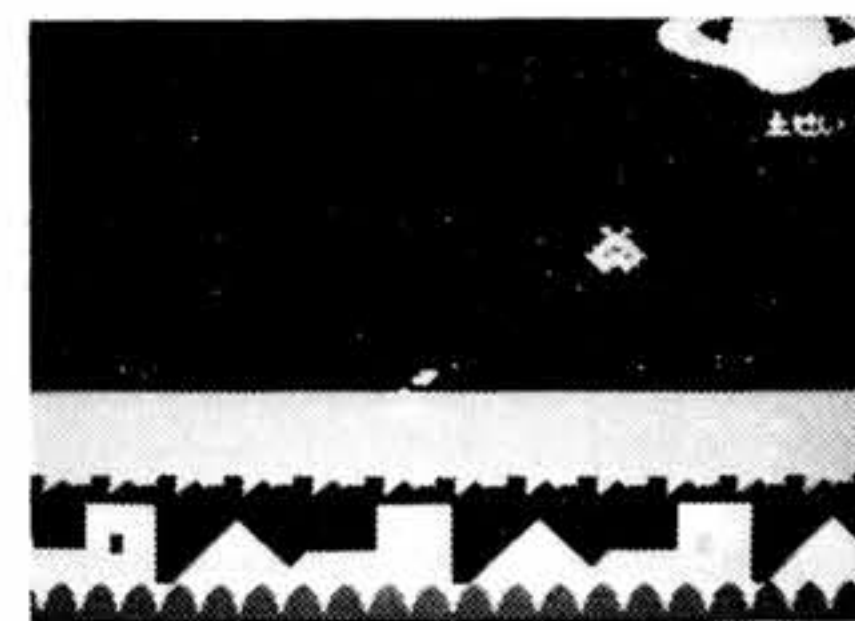
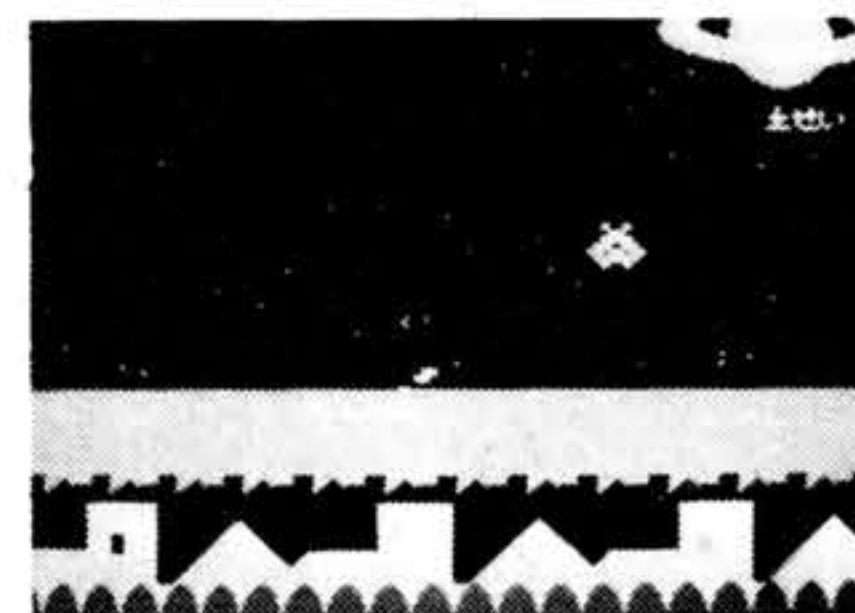
2．1　プログラムの内容

この章では、MSX BASICの代表的な命令語を使ってプログラム作成の練習をします。プログラムは下の写真のような流れで入力していきます。

プログラムを入力しながらいろんな命令の使い方を覚えましょう。また、命令のくわしい説明は第3章を見てね。

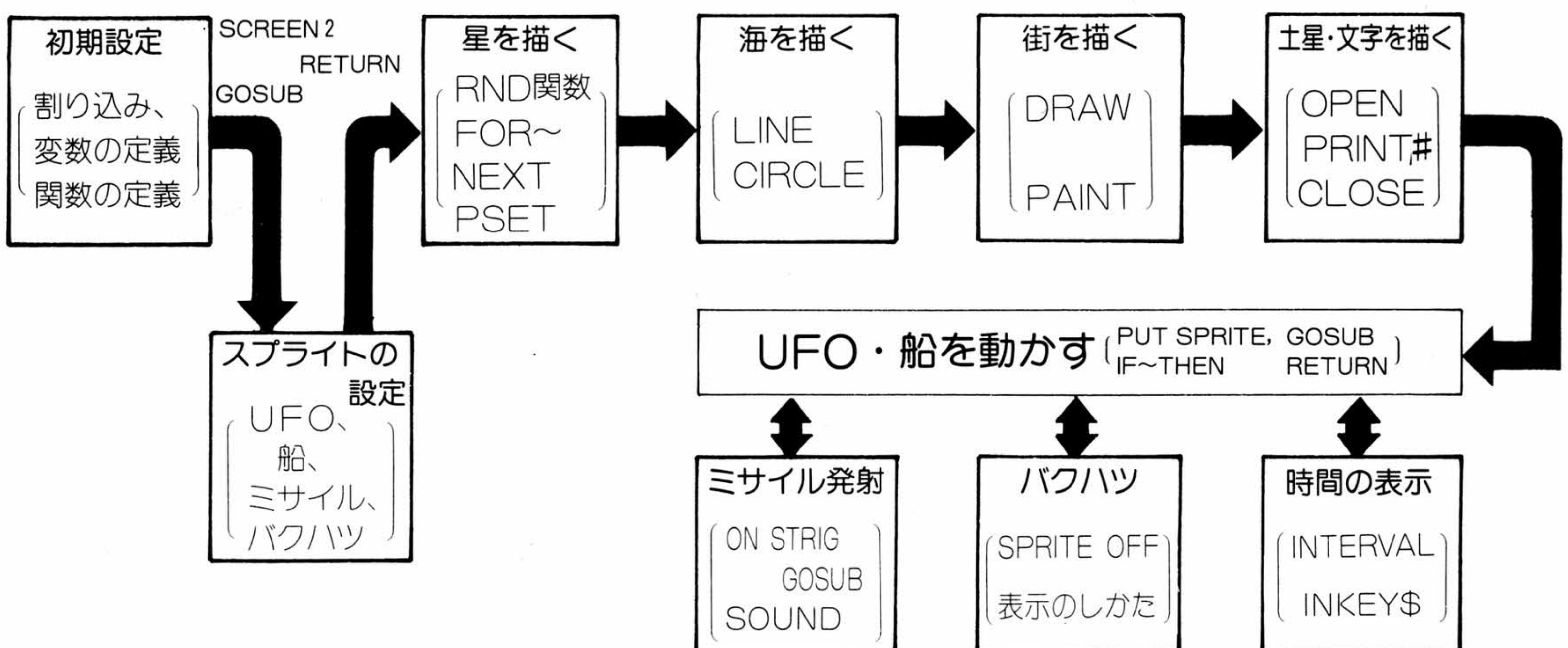
ガンバッテネ！

2．2
空に星を描く

2．3
海を描く

2．4
街を描く

2．5
土星・文字を描く

2．6
スプライトの設定

2．7
UFO・船を動かす

2．8
ミサイルを発射してスコアを表示する

（すべて入力した後プログラムの流れ）

- 初期設定（割り込み、変数の定義、関数の定義） → SCREEN 2 / GOSUB → スプライトの設定（UFO、船、ミサイル、バクハツ） → RETURN
- → 星を描く（RND関数、FOR～NEXT、PSET）
- → 海を描く（LINE、CIRCLE）
- → 街を描く（DRAW、PAINT）
- → 土星・文字を描く（OPEN、PRINT#、CLOSE）
- → UFO・船を動かす（PUT SPRITE，GOSUB、IF～THEN、RETURN）
  - ⇔ ミサイル発射（ON STRIG GOSUB、SOUND）
  - ⇔ バクハツ（SPRITE OFF、表示のしかた）
  - ⇔ 時間の表示（INTERVAL、INKEY$）

2. 2　空に星を描く

空の色を決め、星を90個描きます。また、変数Rの値と関数FNAの式を決めています。（まず、NEW命令を実行して前のプログラムを消してから、入力してください。）

```
50 '**** ﾊｲｹｲ ｦ ｶｸ ****  ←コメント文
60 COLOR 15,1,1:SCREEN 2,1,0  ←空の色と画面モードの指定。
70 R=3.1416/180:DEF FNA(A)=RND(1)*A  ←変数と関数FNAの設定。
80 '** ﾎｼ ｦ ｴｶﾞｸ  ←コメント文
90 FOR S=1 TO 90  ←┐
100 PSET(FNA(255),FNA(191)),FNA(13)+2  ←星を描く。
110 NEXT S  ←┘ 90回くり返す。

800 GOTO 800  ←画面を表示し続ける。
```


40行まで：「2.8 ミサイルを発射しスコアを表示する」で入力します。

50行・80行：あとでプログラムを見直したとき、「ここで……をしています」とわかりやすいようにコメント文をいれます。

60行：COLOR文で画面の色を決め、SCREEN文で画面モードを指定します。

COLOR 15, 1, 1
- 15：文字の色は白色
- 1：空の色は黒色
- 1：画面の周辺色は黒色

SCREEN 2, 1, 0
- 2：グラフィックモード
- 1：8×8の拡大スプライト
- 0：キークリック音を出さない。

70行：Rという変数の値とFNAという関数の式を設定します。

R＝3.1416/180≒0.0175　←CIRCLE文で使います。

DEF FN文により、何回も使う計算式を短かくし、プログラムを簡単にします。

DEF FNA(A)=RND(1)*A
- DEF：関数の設定
- FNA：関数名
- (A)：引数
- RND(1)*A：設定する式

100行：適当な位置に星（点）を描きます。

PSET (FNA (255), FNA (191)), FNA (13)+2
- PSET：点を描く
- FNA (255)：横に0～255の位置
- FNA (191)：縦に0～191の位置
- FNA (13)+2：点の色は黒と透明色を除く14色（2～15）

90行・110行：FOR～NEXT文により、100行のPSET文を90回くり返します。

FOR S=1 TO 90：～100行のプログラム～：NEXT S
- FOR：くり返しの始まり
- S：変数（カウンターとして使います。）
- 1：初めの値
- 90：終りの値
- NEXT S：くり返しの終り

800行：SCREEN 2の画面を見続けるために無限ループを作ります。

以上のプログラムを入力した後、実行（F5 キーを押す）すると、右のような画面になります。

プログラムの実行を止めるときは、CTRL キーを押しながら STOP キーを押します。

2.3　海を描く

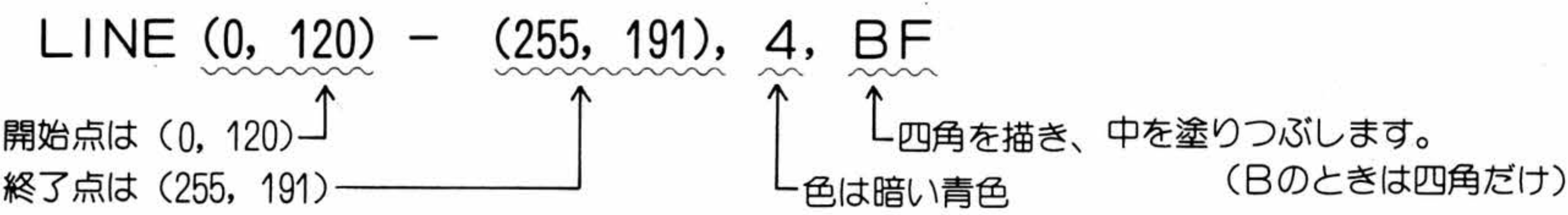
海を青く塗り、波を描きます。

```
120 '** ウミ ヲ エカ゛ク  ←――――― コメント文
130 LINE(0,120)-(255,191),4,BF ←―― 海を描く
140 FOR S=1 TO 20 ←―――――――――┐
150 X=FNA(220)+30:Y=FNA(25)+125 ←―― 波の位置を決める
160 CIRCLE(X,Y),3,15,R*45,R*135 ←―― 波(白い弧)を描く
170 NEXT S ――――――――――――――――┘
                                    20回波を描く
```

120行：「海を描いています」という意味のコメント文を入力します。

130行：LINE文で海を青く塗ります。（青い四角を描く）

LINE (0, 120) － (255, 191), 4, BF

- 開始点は（0, 120）
- 終了点は（255, 191）
- 色は暗い青色
- 四角を描き、中を塗りつぶします。（Bのときは四角だけ）

140行～170行：150行と160行の命令（波の位置を決め、白い弧を描く）を20回行ないます。

150行：FNA関数により波の位置を適当に決めます。
FNA関数の内容は70行で設定されたものです。

用　語

■ラジアン

パソコン内では、CIRCLE命令の角度をラジアンという単位で計算しています。しかし、ラジアンという単位では一般にわかりにくいので、あらかじめ変数Rに１度あたりのラジアンの値を代入しておきます。

R＝3.1416/180≒0.0175

これで以後のCIRCLE文などで角度を指定するとき、Rに度単位の角度をかけ算すれば、ラジアン単位の角度となります。

例　　30°は　R＊30 ラジアン

ラジアンと度は、次のように対応しています。

ラジアン	0	π/6	π/4	π/2	π	3π/2	2π
度	0°	30°	45°	90°	180°	270°	360°

π(パイ)＝3.14159265358979323…
省略して 3.1416

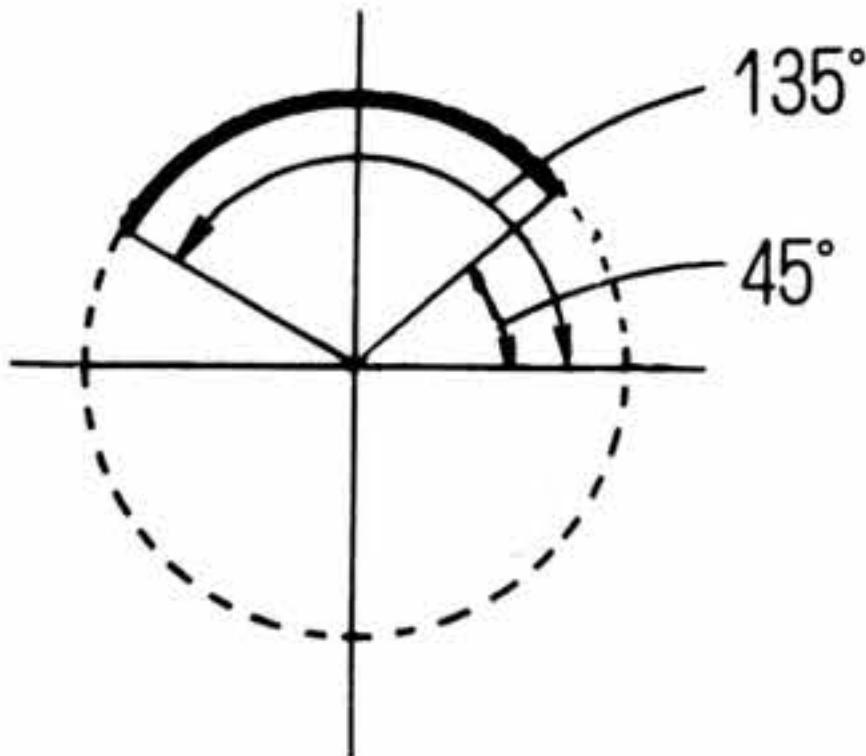
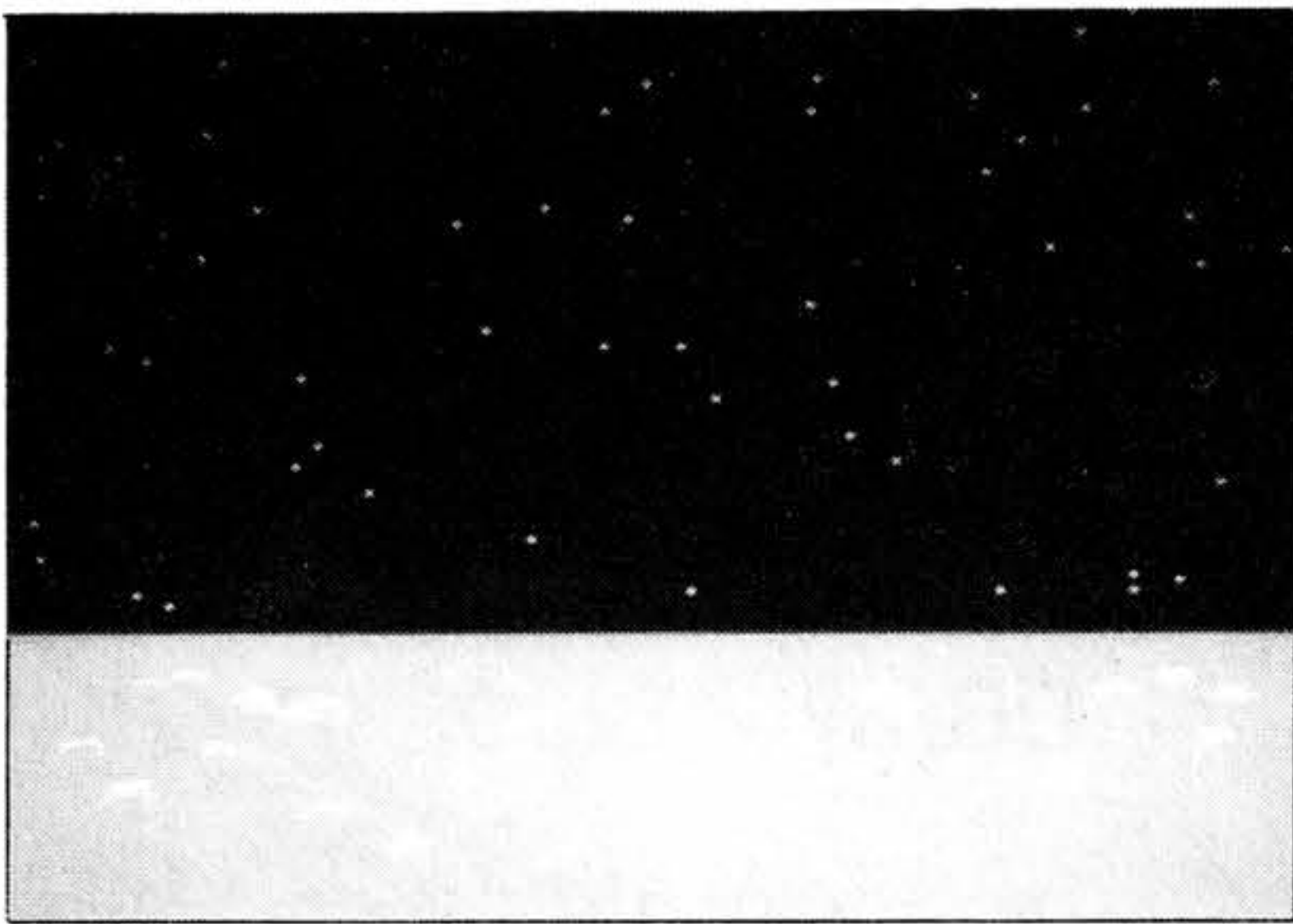
■乱数（RND関数）

RND(１)はサイコロを転がすように０～１の適当な値を作ります。
RND関数は０～１の数(0,0.1,0.2…,…,1)しか決められないので、画面の縦横の大きさに合せて何倍かにしています。

例　　FNA(255)＝RND(１)＊255…　0～255の数

X＝FNA（220）＋30：Y＝FNA（25）＋125

160行：CIRCLE文で白い波を描きます。（変数Rは70行で設定した値です）

CIRCLE（X，Y），3，15，R＊45，R＊135

以上のプログラムを入力すると、右の図のような画面となります。
（止めるときは、CTRL＋STOP）

2.4　街を描く

DRAW命令で街並みを描きます。その後、CIRCLE命令で木を描きます。

```
180 '** ﾏﾁ ｦ ｴｶﾞｸ                       ← コメント文
190 D$="E5R20U15R20D25R5E20F15"          ← DRAW命令で使う文字列を作る
200 DRAW "BM0,150S1C1"                   ← 遠い街並の始まりの位置
210 FOR S=1 TO 13:DRAW"XD$;":NEXT S      ← 遠い街並を描く
220 PAINT(200,155),1                     ← 黒く塗る
230 DRAW "BM0,175S4C14"                  ← 近い街並の始まりの位置
240 FOR S=1 TO 3:DRAW"XD$;":NEXT S       ← 近い街並を描く
250 PAINT(250,180),14                    ← 灰色を塗る
260 LINE(32,165)-(35,170),1,BF           ┐
270 LINE(114,160)-(116,165),8,BF         ├ 窓を描きます
280 LINE(200,165)-(204,170),8,BF         ┘
290 '** ｷ ｦ ｴｶﾞｸ                        ← コメント文
300 FOR X=5 TO 260 STEP 12               ┐
310 CIRCLE(X,195),15,12,30*R,150*R,2.5   ├ 縦長の円を描く
320 NEXT X                               ┘
330 PAINT(255,185),12                    ← 木を濃い緑で塗ります
```

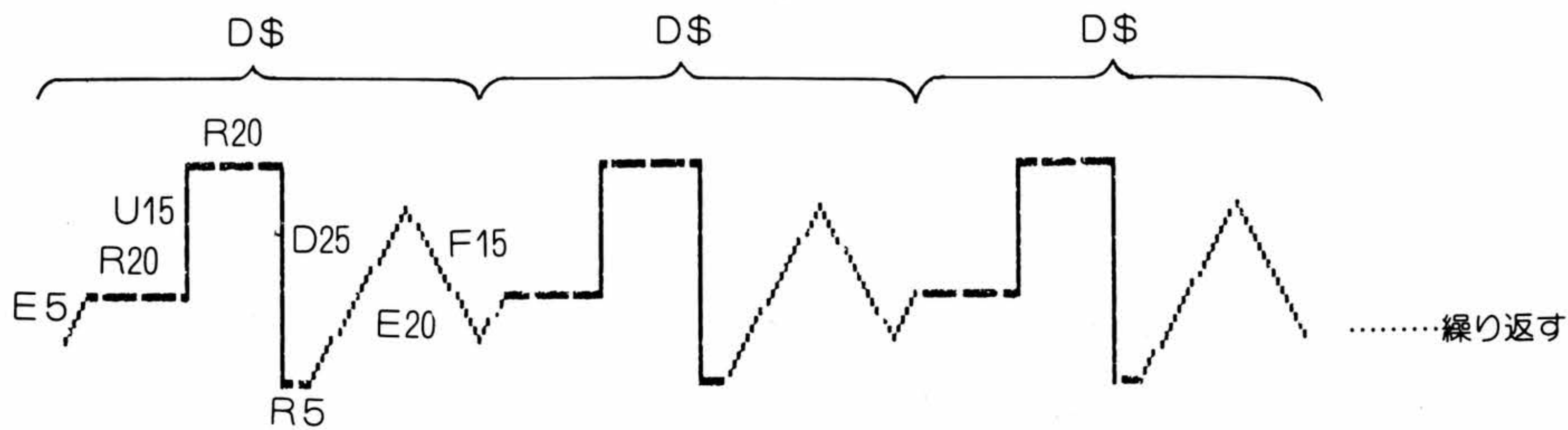
180行：コメント文です。

190行：210行と240行で使うDRAW命令の文字列のデータ（街並みの形をあらわす）を作ります。各々の文字のもつ意味については第３章をご覧ください。

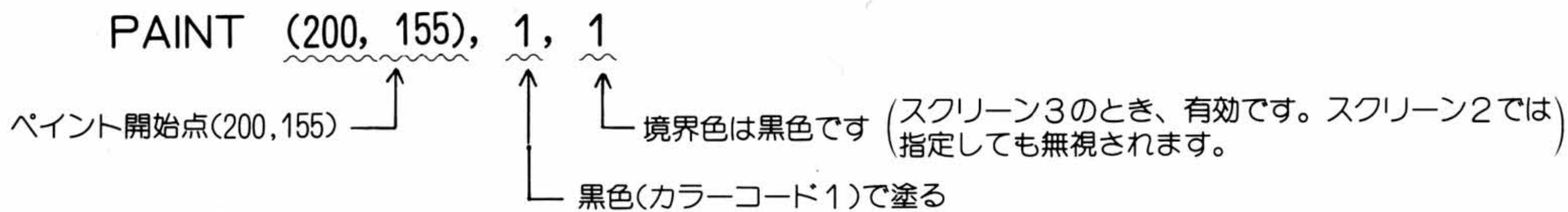
200行：210行で描く遠い街並の開始位置を決めています。

DRAW "BM0，150 S1 C1"

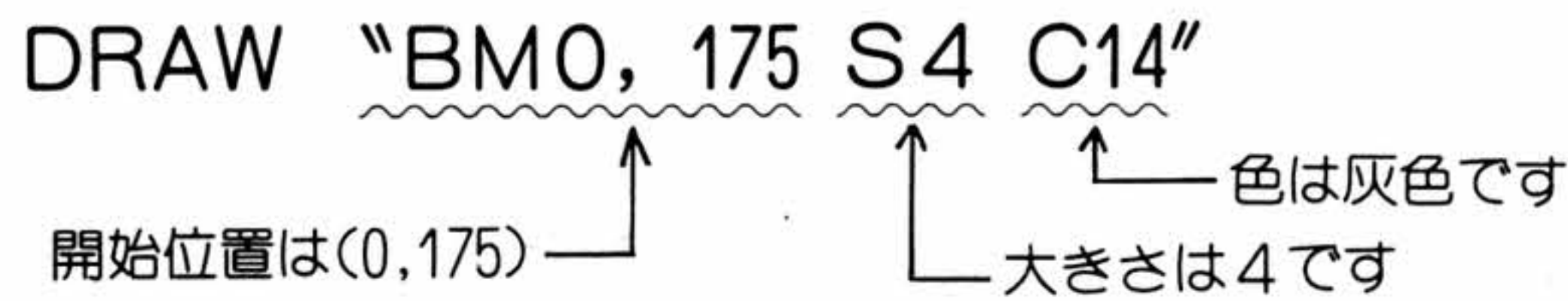
190行，210行で決めた文字列のデータに従って、線(遠い街並み)を描きます。

D$ D$ D$
R20
U15
D25
F15
R20
E5
E20
R5
………繰り返す

220行：遠い街並を黒色で塗りつぶします。PAINT文は、次のように設定します。

PAINT (200, 155), 1, 1

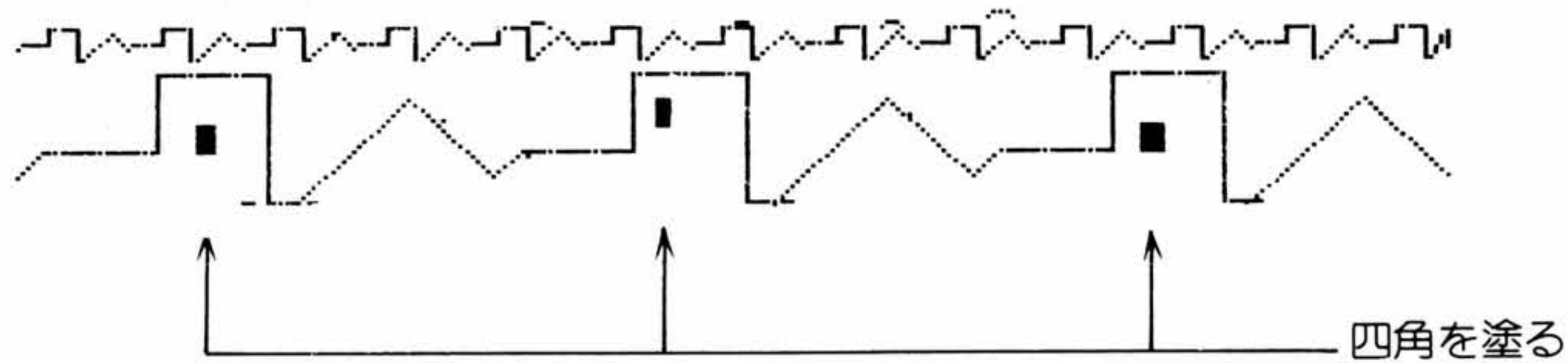
ペイント開始点(200,155)
境界色は黒色です (スクリーン3のとき、有効です。スクリーン2では指定しても無視されます。)
黒色(カラーコード1)で塗る

230行：240行で描く近い街並の開始位置を決めています。

DRAW "BM0, 175 S4 C14"

開始位置は(0,175)
大きさは4です
色は灰色です

240行：190行で決めた文字列のデータに従って線(近い街並み)を描きます。

250行：近い街並を灰色で塗りつぶします。

260行〜280行：近い街並に窓（塗りつぶした四角）を描きます。

四角を塗る

290行：コメント文です。

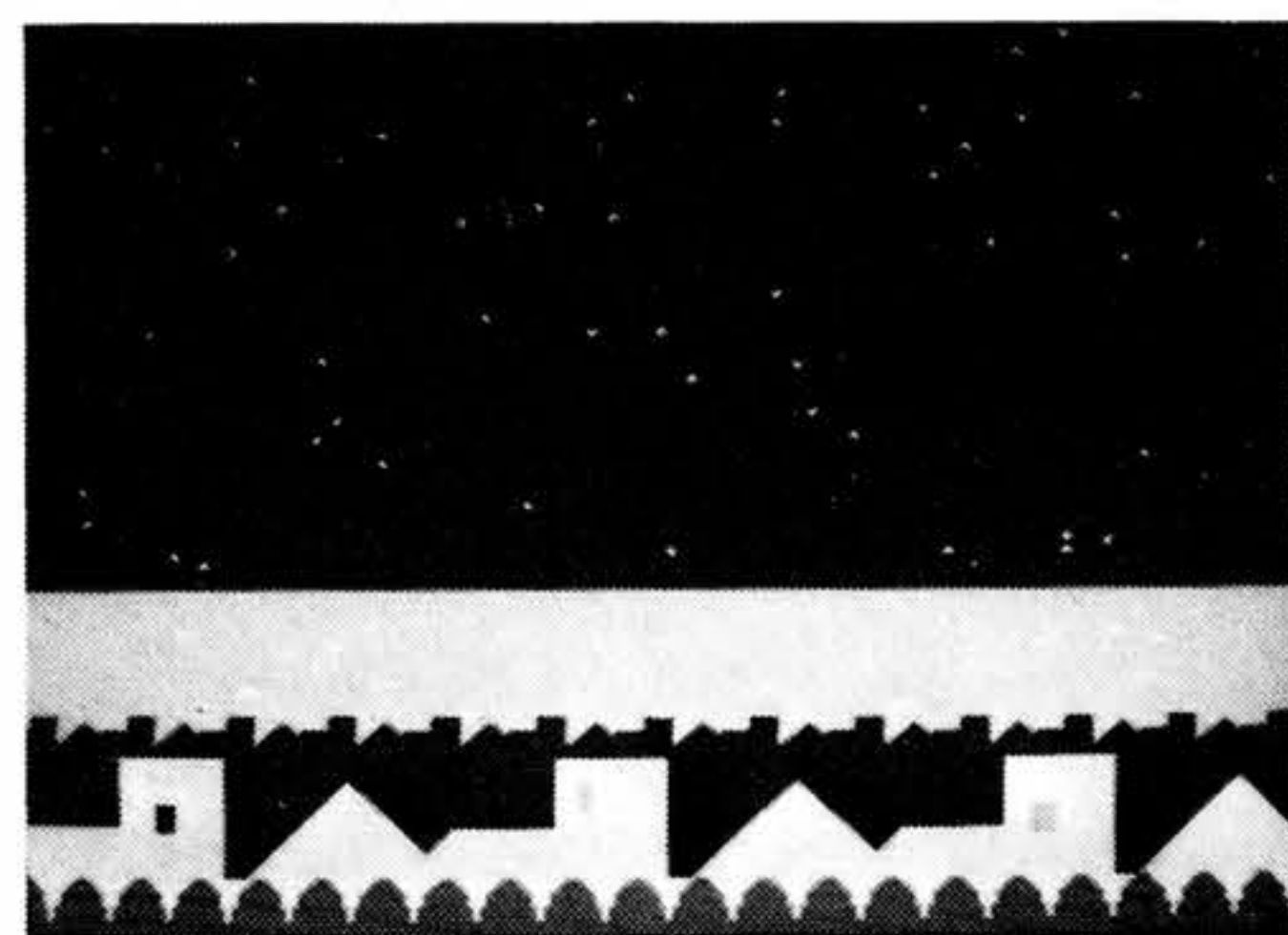
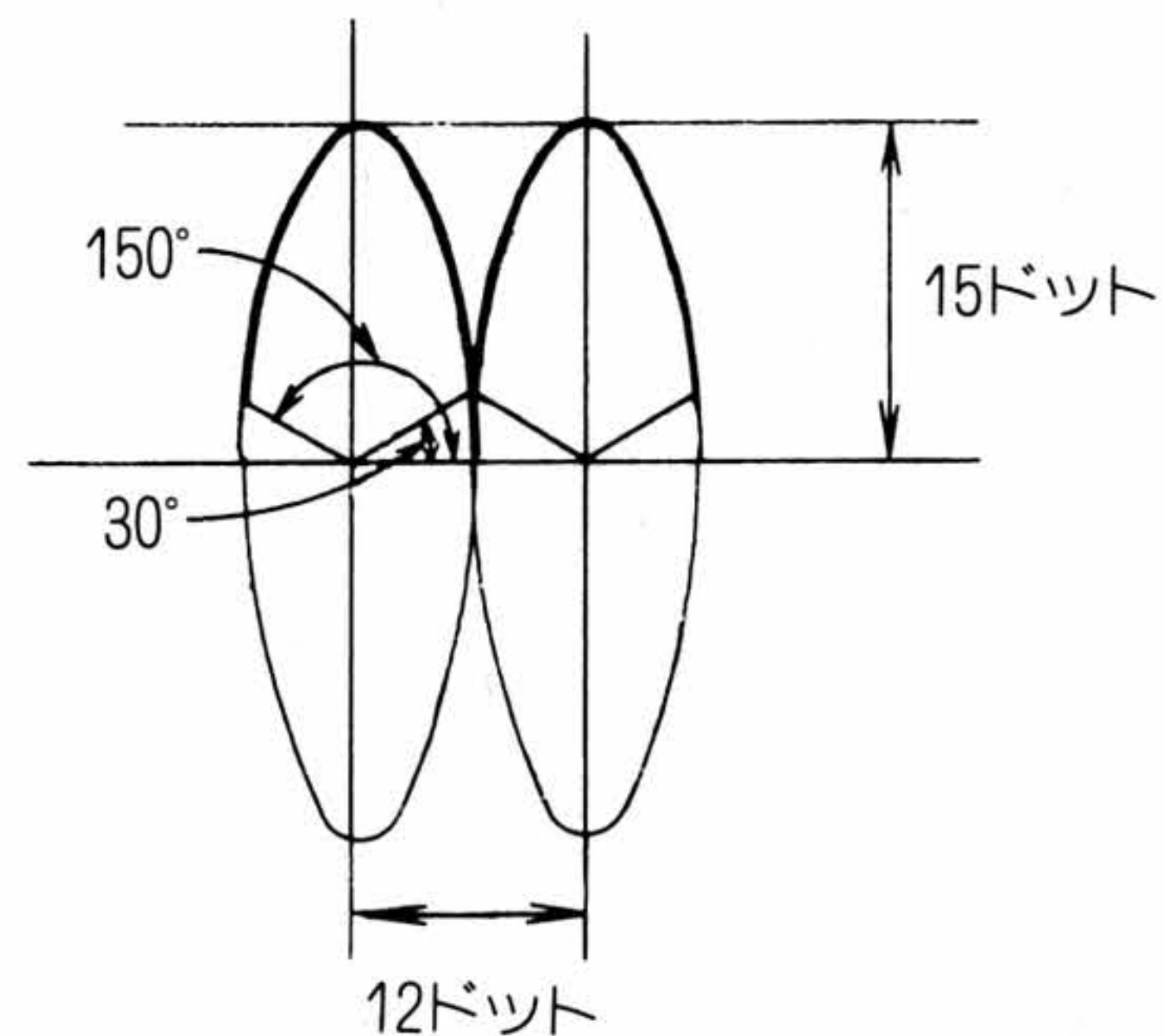
300行〜320行：CIRCLE文で描く円の中心を変化させ、半円を約22個描きます。

FOR〜NEXT文のステップが12ドットとなっているのは、比率が2.5のためです。

15÷2.5＝6…半径　6×2＝12…中心間の距離

150°
30°
15ドット
12ドット

330行：PAINT文により、木を濃い緑で塗ります。

以上のプログラムを入力すると、右の図のような画面となります。

2.5 土星・文字を描く

CIRCLE命令で土星を描いた後、OPEN、PRINT＃命令で文字を表示します。

```
340 '** 土セイ ←―――――――――――――――――――――― コメント文
350 CIRCLE(224,8),15,10 ←―――――――――――――― 土星を描きます
360 PAINT(224,8),10 ←
370 CIRCLE(224,4),25,14,R*120,R*50,.25
380 CIRCLE(224,6),35,14,R*110,R*30,.3 ←―――― 土星の輪を描きます
390 PAINT(224,15),14 ←
400 '** モジ゛ ヲ カク ←―――――――――――――――――― コメント文
410 OPEN "GRP:" FOR OUTPUT AS #1 ←――――――― グラフィック画面のファイルを開きます
420 DRAW "BM220,30":PRINT #1,"土せい" ←――――― 「土せい」を表示します
430 DRAW "BM16,6":PRINT #1,"SCORE HI-SCO ― 「SCORE」と「HI-SCORE」を表示
RE"                                            します。
440 DRAW "BM16,14":PRINT #1,0 ←―――――――――― 「0」を表示します
450 DRAW "BM16,26":PRINT #1,"TIME" ←――――――― 「TIME」を表示します
```

●星を描く

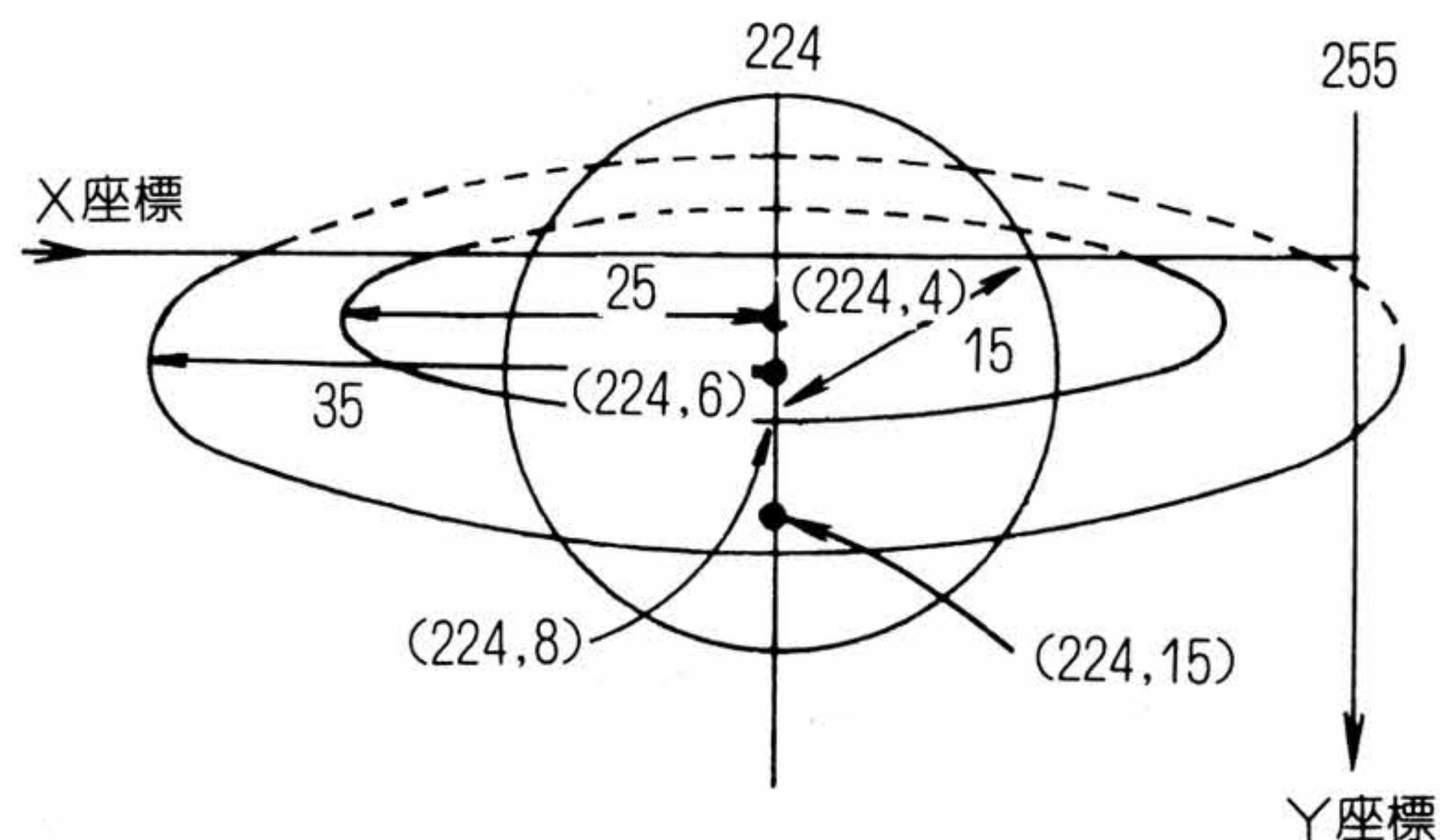
340行：コメント文です。

350行・360行：土星の円を描き、黄色(カラーコード10)で塗ります。指示している座標は図のとおりです。

370行〜390行：土星の輪を描き、灰色（カラーコード14）で塗ります。

CIRCLE文、PAINT文の座標は図のように近い値なので打ち間違いのないように注意してください。
ところで、この土星はなぜ画面の端にいるのでしょう。もっと中へ入らないのでしょうか？

そう思った人は、350行から390行のプログラムの代りに次のプログラムを入力してください。


```
340 ' 土セイ
350 CIRCLE(204,18),15,10
360 PAINT(204,18),10
370 CIRCLE(204,14),25,14,,,.25
380 CIRCLE(204,16),35,14,,,.3
390 PAINT(204,25),14
395 CIRCLE(204,18),15,10,,R*180
397 PAINT(204, 7),10
```

入力した人は理由がわかったと思います。そうです、土星の輪が星の上に重ってしまうからなのです。そのため395行、397行で、星の塗り直しをしています。

●文字を書く

400行：コメント文です。

410行：グラフィック画面に文字や記号を表示できるよう、ファイルを開きます。

OPEN "GRP:" FOR OUTPUT AS #1

- "GRP:" ← グラフィック画面のファイルを指定
- FOR OUTPUT ← ファイルへ出力することを宣言します
- AS #1 ← ファイル番号として1を指定します（1～15まで指定できます）

これより以後はPRINT #文で文字の表示ができます。

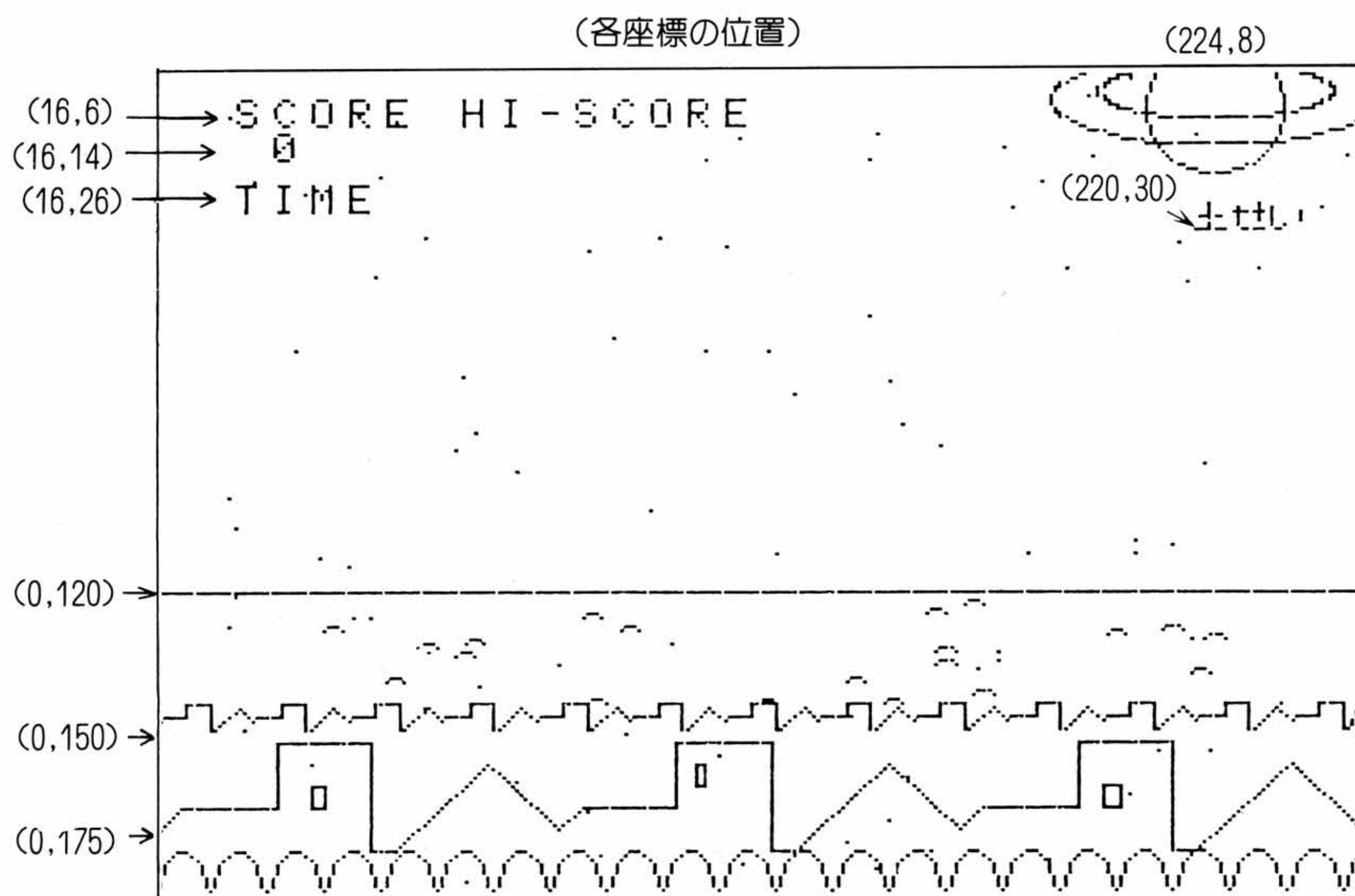
420行～450行：DRAW文により表示の位置を決め、PRINT #文により文字・記号を表示します。

PRINT #文はファイルに出力する命令で、PRINT #(ファイル番号)に続く文字・記号がグラフィック画面に表示されますが、その表示位置はLP(ラストポイント：最終参照点)です。このためDRAW文でLPを変えて、表示位置を設定します。(「2.4.街を描く」で使ったDRAW文も同じ使い方です)

例 DRAW "BM220,30":PRINT # 1, "土せい"

- DRAW "BM220,30" ← 表示の開始位置は(220,30)
- PRINT # ← ファイル番号に続く文字・記号を表示する
- 1 ← ファイル番号は1(OPEN文と同じ番号)
- "土せい" ← 表示する文字

以上のプログラムが無事入力できると、右の図のような画面となります。
(止めるときは[CTRL]+[STOP])

(各座標の位置)

2.6 スプライトの設定

画面がだいたいできあがりましたので、UFO、船などのスプライトを作りましょう。
ここではスプライトのパターン設定にREAD、DATA文を使います。この方法は、プログラムの追加、変更が手軽にできます。
作るパターンは、「UFO」、「船」、「ミサイル」、「爆発(バクハツ)」の4つです。(なお、このプログラムだけではスプライトパターンは表示されません。次の2-7項のプログラムも入力してください。)

```
  60 COLOR 15,1,1:SCREEN 2,1,0:GOSUB 1030  ← GOSUB文の追加

1030 '**** ｽﾌﾟﾗｲﾄ ｾｯﾃｲ ****                 ← コメント文
1040 FOR T=4 TO 7:A$=""
1050 FOR K=1 TO 8
1060 READ B$:A$=A$+CHR$(VAL("&H"+B$))    ← パターンデータの読み込み
1070 NEXT K
1080 SPRITE$(T)=A$                        ← スプライトの設定
1090 NEXT T:RETURN                        ← くり返し、RETURN
1100 ' UFO ﾊﾟﾀｰﾝ
1110 DATA 24,18,3C,66,DB,7E,24,00          ← 「UFO」のデータ
1120  ' ﾌﾈ  ﾊﾟﾀｰﾝ
1130 DATA 03,06,00,10,10,38,EE,7C          ← 「船」のデータ
1140 'ﾐｻｲﾙ ﾊﾟﾀｰﾝ
1150 DATA 10,10,00,00,00,00,00,00          ← 「ミサイル」のデータ
1160 ' ﾊﾞｸﾊﾂ ﾊﾟﾀｰﾝ
1170 DATA 91,4A,84,11,54,09,22,08          ← 「バクハツのデータ」
```

60行:60行にGOSUB命令を追加してプログラムの流れを1030行へ移します。

1030行:コメント文です。GOSUB命令の行先指定番地となっています。

1040行:FOR~NEXT文によりパターンを4個設定します。
A$="" 文により文字変数A$をヌルストリングにします。これは、A$を4つのパターン設定にくり返し使うため、前に設定した内容が残っていると後のパターンに影響するので、一度A$の内容を消します。

1050行・1070行:FOR~NEXT文によりデータを8回読みます。(8×8のパターンのため)。

1060行:スプライトのパターンデータを文字変数A$に置き換えます。
文字変数B$にデータを読み込み、この行で16進数に変換します。そして、そのキャラクタデータをA$に加えます。

READ B$: A$=A$+CHR$(VAL("&H"+B$))

- B$:データ10,30などを文字列として読み込みます。
- A$=A$+:文字列A$に、データを加えていきます
- CHR$:数値に対応するキャラクターを得ます。
- VAL:文字列を数値に変換する "&H"+"10"→&H10
- "&H"+B$:データを&H10などの16進数の文字列とする

1080行:A$をスプライトパターンとして設定します。

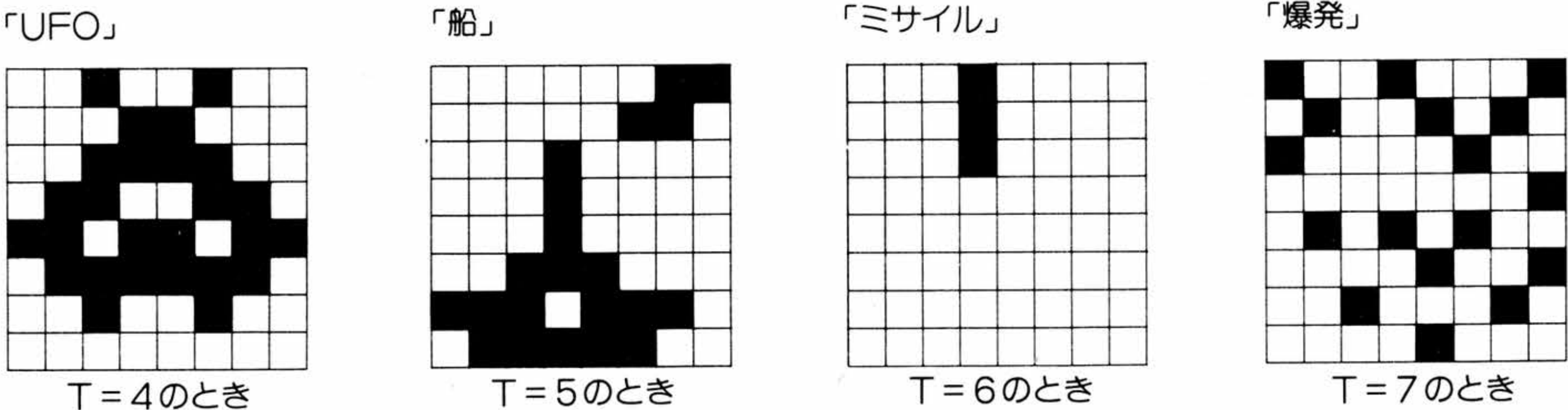
SPRITE$(T)=A$

- T:スプライトパターン番号
- A$:スプライトのパターンデータ

1090行：4個のパターンを設定し、70行へ実行を移します。

1100行〜1170行：スプライトのパターンは、図のように設定されます。

「UFO」　「船」　「ミサイル」　「爆発」

T＝4のとき　T＝5のとき　T＝6のとき　T＝7のとき

2.7 UFO・船を動かす

UFOが画面の右から左へ揺れながら動きます。また、カーソルキーで船を左右へ移動させます。

```
460 '**** UFO ヲ ウゴカス ****            ← コメント文
470 TIME=0:SC=0:CO=15:XX=240            ← 時間、スコア、色などの設定
480 TI=20:YY=208:STRIG(0) ON
490 FOR K=240 TO 10 STEP -3             ← FOR～NEXT文で「UFO」が動く
500 Y=FNA(5 )+30+K/6                    ← 「UFO」のY座標
510 X=FNA( 5)+K                         ← 「UFO」のX座標
520 SPRITE ON:INTERVAL ON               ← 割り込みの許可
530 PUT SPRITE 4,(X,Y),CO,4             ← 「UFO」の表示
540 GOSUB 600                           ← キー入力チェックプログラムへ
550 IF MF=1 THEN YY=YY-5:IF YY<=0 THEN M
F=0:YY=208                              ← ミサイルを発射したときの処理
560 IF(K MOD 10)=0 THEN PLAY"L64O5DC"   ← 「UFO」の動く音
570 PUT SPRITE 6,(XX,YY),8,6            ← 「船」と「ミサイルの表示」
580 PUT SPRITE 5,(XX,184),15,5
590 NEXT K:GOTO 490                     → くり返し
600 '**** key チェック ****
610 A=STICK(0)                          ← カーソルキーのチェック
620 IF A=3 THEN XX=XX+5:IF XX>240 THEN X ← 「船」を右へ移動
X=240
630 IF A=7 THEN XX=XX-5:IF XX<5 THEN XX= ← 「船」を左へ移動
5
640 RETURN                              ← キー入力チェックの終了
```

460行：コメント文です。

470行・480行：時間(TIME)、スコア(SC)、UFOの初めの色(CO)、船の初めの位置(XX、YY)、ゲーム時間(TI)を設定します。
また、STRIG(0)ONで、スペースキーの割り込みを許可します。( )内の0は、スペースキーを意味します。割り込み処理ルーチンは、次の項で設定します。

490行
・
590行：FOR～NEXT文により「UFO」を画面の右から左へ移動させます。

500行
・
510行：「UFO」のX座標とY座標を計算します。計算式を変えることによってUFOの動きが変わります。(ただし、「UFO」が船とぶつからないような計算式にしてください。)

520行：スプライトの衝突割り込みと、タイマー割り込みを許可します。割り込み処理ルーチンは、次の項で設定します。

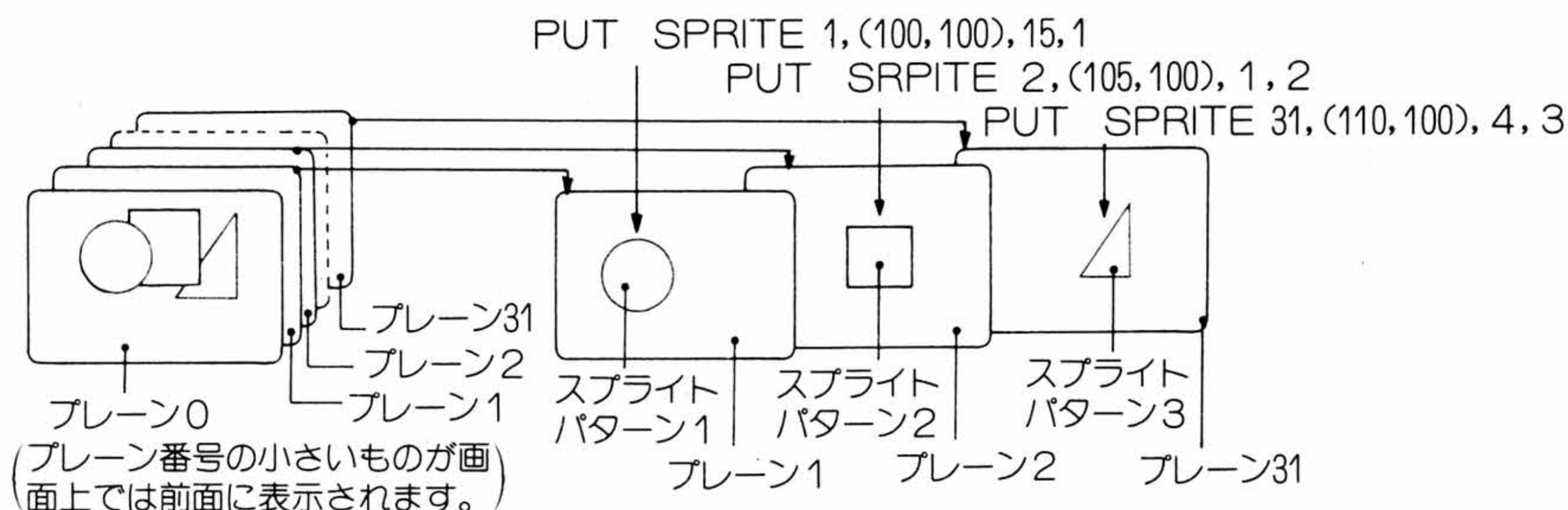
530行：「UFO」のスプライトをプレーン番号4に設定します。

PUT SPRITE 4, (X, Y), CO, 4

スプライトパターンはSPRITE$命令で設定され、スプライトパターンの表示位置はPUT SPRITE命令で指定されます。

例　パターンの設定　SPRITE$(m)="パターンのデータ"

パターンを表示する位置の設定　PUT SPRITE n, (X, Y), c, m

スプライトの設定例

540行：カーソルキーの入力チェックプログラムへ移ります。

550行：「ミサイル」が発射されているとき(MF＝1のとき)、「ミサイル」を5ドット上へ上げます。そして「ミサイル」が「UFO」に当らず画面の外に出たとき、MF＝0、YY＝208としてミサイルを消します。

560行：「UFO」が10回動くと1回、音を出します。ミュージックマクロ命令については、第3章のPLAYをご覧ください。

PLAY"<L6405DC>"

570行・580行：「ミサイル」を赤色で、「船」を白色で表わします。「ミサイル」は発射されていないときはYY=208なので、画面に表示されません。

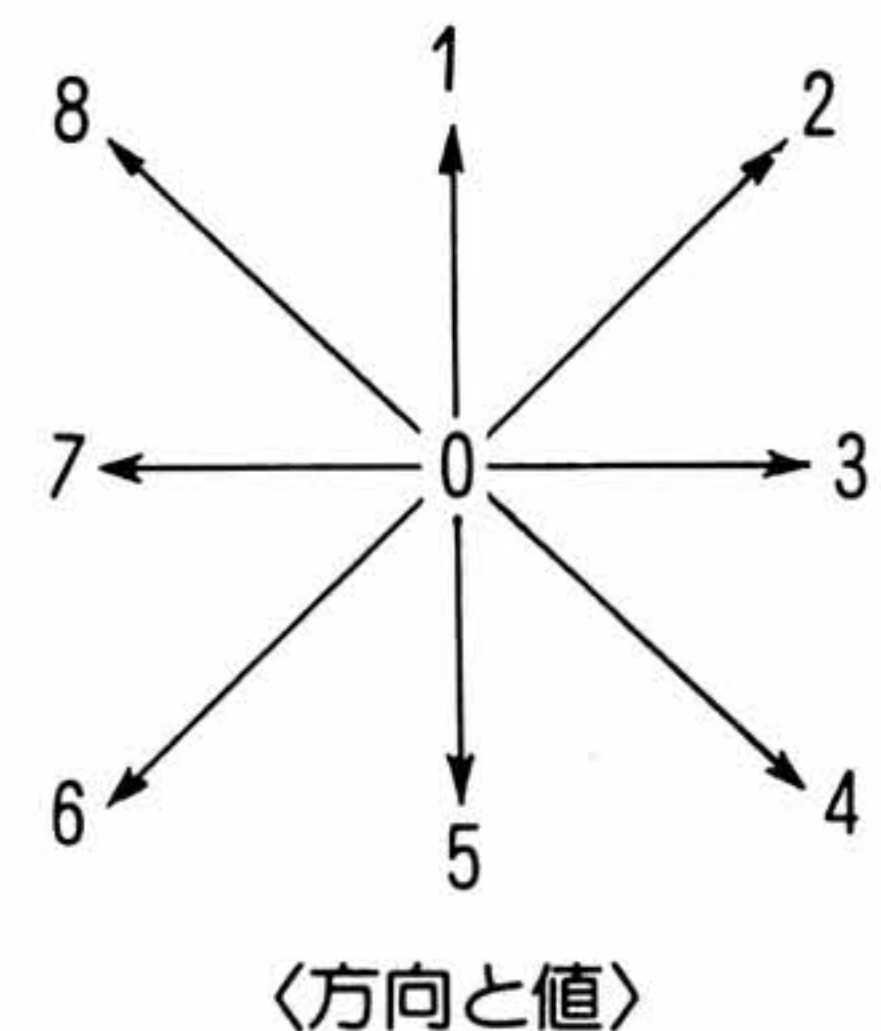
600行：コメント文です。540行のGOSUBの行先指定番地です。

610行：STICK関数を使ってカーソルキーの状態をチェックします。
STICK関数はジョイスティックのレバー（またはキーボードのカーソルキー）の方向によって0～8の値を得ます。どの方向も指定しないときは0です。

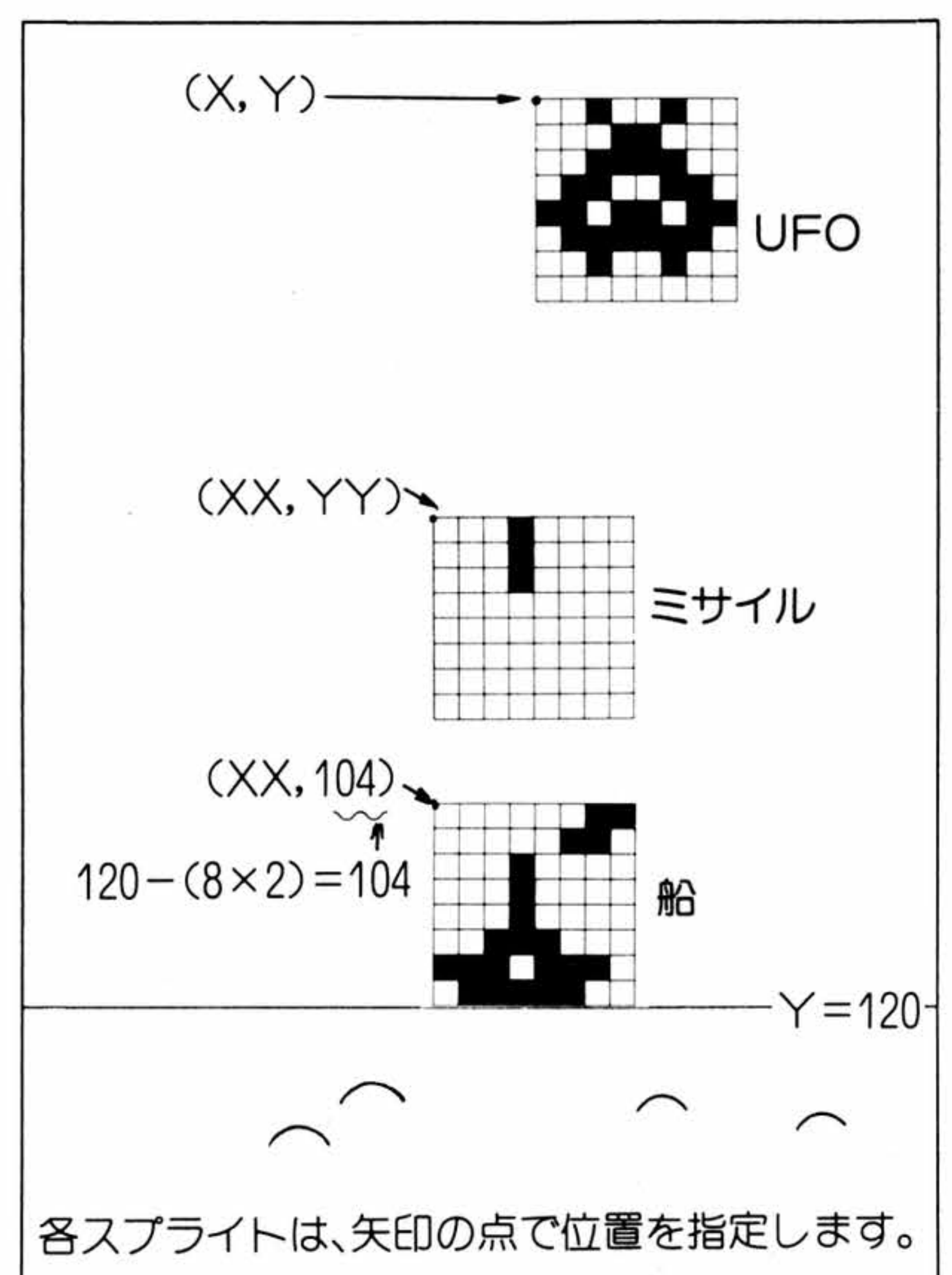
A=STICK（0）

- STICK：0～8の値になります。
- （0）：0のとき、カーソルキーをチェック / 1のときジョイスティック1をチェック / 2のときジョイスティック2をチェック

〈方向と値〉

STICK関数の（　）の数字は、今は、カーソルキーで動くよう、0にしています。ジョイスティックを使うときは、1または、2に変えてください。
また、A=STICK(0)の変数Aにより、STICK関数の値を記憶します。

620行・630行：キーの押されている方向によって船のX座標(XX)を計算しています。
ここで、XXが240より大きいときは5に、5より小さいときは240に設定しています。

各スプライトは、矢印の点で位置を指定します。

以上のプログラムで、SCREEN　2での画面が続きます。
(エラーがあったり、CTRL + STOP しないかぎり)。
RUN リターン することで、図のような画面となります。

2.8 ミサイルを発射しスコアを表示する

割り込み処理ルーチンの開始行番号を設定し、各処理ルーチンを作ります。
処理ルーチンは、ミサイル発射処理、爆発処理、タイマー割込み処理です。

```
 20 ON SPRITE GOSUB 720 ←── スプライトがぶつかったときの処理を設定
 30 ON STRIG GOSUB 650 ←── スペースキーを押したときの処理を設定
 40 ON INTERVAL=60  GOSUB 840 ←── 1秒ごとの割り込み処理を設定

650 '**** ﾐｻｲﾙ ｦ ｳﾂ **** ←── ミサイル発射処理の開始
660 MF=1:YY=90 ←── ミサイルフラグの設定
670 SOUND 2,0:SOUND 3,0 ←─┐
680 SOUND 6,2:SOUND 7,46 ←─┤ 発射音を出します
690 SOUND 9,16:SOUND 11,50 ←─┤
700 SOUND 12,2:SOUND 13,0 ←─┘
710 RETURN ←── 処理を終了します
720 '**** ﾊﾞｸﾊﾂ ｼﾏｽ **** ←── 爆発処理の開始
730 SPRITE OFF:SC=SC+10:MF=0 ←── スコア計算、ミサイル、フラグの取り消し
740 PUT SPRITE 1,(XX,YY-4),8,7 ←── 「バクハツ」スプライトの表示
750 PUT SPRITE 6,(XX,209),0,6
760 SOUND 9,16:SOUND 11,50 ←─┐ 爆発音を出します
770 SOUND 12,50:SOUND 13,0 ←─┘
780 FOR R=0 TO 20:NEXT R
790 PUT SPRITE 1,(10,209),0,7 ←── 「バクハツ」スプライトを消します
800 CO=FNA(13)+2:YY=208 ←── 「UFO」の色の計算
810 LINE(13,13)-(55,21),1,BF ←─┐ スコアの表示
820 DRAW"BM16,14":PRINT #1,SC ←─┘
830 RETURN ←── 処理を終了します
840 '**** ﾀｲﾏｰ **** ←── タイマー割込み処理の開始
850 SPRITE OFF:INTERVAL OFF
860 TI=TI-1:IF TI<0 THEN TI=0 ←── 時間の計算
870 LINE(16,33)-(45,41),1,BF ←─┐ 時間の表示
880 DRAW "BM 16,35":PRINT #1,TI ←─┘
890 IF TI>0 THEN RETURN '** ﾓﾄﾞﾙ ←── 処理の終了
900 IF SC>HSC THEN HSC=SC ←─┐ スコアの計算と表示    時間が0のとき
910 LINE(64,13)-(116,21),1,BF ←─┤                    の処理ルーチン
920 DRAW "BM 72,14":PRINT #1,HSC ←─┘
930 STRIG(0) OFF:COLOR 15:GOSUB 990 ←── 「ゲームオーバー」の表示へ
940 IF INKEY$=CHR$(13) THEN 950 ELSE 940 ←── キーチェック
950 COLOR 1:GOSUB 990:COLOR 15 ←─┐ ゲーム再開
960 LINE(13,13)-(55,21),1,BF ←─┤
970 DRAW"BM16,14":PRINT #1,0 ←─┤
980 MF=0:RETURN 470 ←─┘
990 '** GAME OVER ←─┐ 「ゲームオーバー」の表示
1000 DRAW"BM60,40":PRINT #1,"GAME OVER !" │
1010 DRAW"BM60,50":PRINT #1,"Push RETURN ←─┤
 Key!"                                      │
1020 RETURN ←─┘
```

● 割り込み設定

20行：スプライトが重なったとき（衝突したとき）にサブルーチンへ実行を移す割り込みを設定します。（割り込みの許可は520行でしています）

ON SPRITE GOSUB　720 ←サブルーチンを開始する行番号（爆発を表示するプログラム）

↑「スプライトが重なったとき、サブルーチンへ行きます。」

SPRITE
- ON…………割込みを許可します
- OFF………割込みを禁止します
- STOP……割込みを一時保留します

30行：スペースキー（またはジョイスティックのトリガーボタン）を押したときにサブルーチンへ実行を移す割り込みを設定します（割り込みの許可は520行でしています）。

ON STRIG GOSUB　650 ←サブルーチンを開始する行番号（ミサイル発射と音を出します）

↑「スペース（トリガーボタン）を押したときサブルーチンへ行きます。」

STRIG（0）ON　………割り込みを許可します。

↑スペースキーの割り込みを意味します。

40行：1秒ごとの割込み（1秒＝60）を設定します。

ON INTERVAL＝60　GOSOB　840

↑割り込みの時間 1秒は60　　↑サブルーチンを開始する行番号

● ミサイル発射処理ルーチン

660行：「ミサイル」の発射を意味する変数（MF）を1にします。このMFが1のとき「ミサイル」が発射されているとみなします。このように、条件判断に使う情報をもった変数を**「フラグ」**といいます。

670行・700行：SOUND文によって「ミサイル」発射の音を出します。SOUND文の値については第3章をご覧ください。

710行：処理を終了して、割り込みの発生したところへもどります。

ちょっと一言

■ミサイルの再発射について

このプログラムでは、このゲームをおもしろくするためにミサイルの再発射ができるようにしてあります。（また、ミサイルの横方向の動きもできるようにしています。）
再発射をしないときは、550行、660行、830行に命令を追加してください。

```
550 IF MF=1 THEN YY=YY-5:IF YY<=0 THEN M
F=0:YY=208:STRIG(0) ON
660 MF=1:YY=90:STRIG(0) OFF
830 STRIG(0) ON:RETURN
```

●爆発処理ルーチン

730行：スプライトの割り込みを禁止します。これは、「UFO」のスプライトの上に「バクハツ」のスプライトを重ねることで割り込みが発生するのを防ぐためです。

スコア(変数SC)を加えます。また、「ミサイル」を消すためMFを0にします。

740行・750行：「バクハツ」のスプライトをプレーン番号1に赤色で表示します。パターンのY座標をYY-4としているのは、ミサイルの当ったところをパターンの中心にするためです。

760行・780行：爆発音を出します。ノイズの周波数を変えることで「ミサイル」発射の音と区別しています。

FOR～NEXT文は、「バクハツ」のスプライトを表示し続けるための待ち時間です。

790行：「バクハツ」のスプライトを消します。

800行：「ミサイル」が当ったことを示すため「UFO」の色を変えます。色は2～15までの14色のいずれかです。また、「ミサイル」を消すためYYを画面の外にします。

810行・820行：スコア表示の部分を黒い四角で塗りつぶして、前のスコアを消します。
また、新しいスコアをPRINT＃文を使って表示します。

830行：すべての処理を終って割り込みの発生したところへもどります。

ちょっと一言

ミサイルの動き

このプログラムでは、簡単にするためにミサイルは1個しか発射できません。何個も発射したい人は、自分で考えてみてください。方法は何通りもあります。(ヒント：配列を使う)。

また、発射したミサイルが船といっしょに動くのはいやだと思う人は、ミサイルのX座標にもう1つ変数を使うと、発射したミサイルはそのまま上昇していきます。(このときは、570行、660行、740行、750行も変えてください。)

●タイマー割り込み処理ルーチン

850行：設定している2つの割り込み(スプライトとタイマー)を禁止します。
これは、他の割り込みによって時間表示の位置がズレるのを防ぐためです。

860行：残り時間を計算します。

870行・880行：時間表示の部分を黒い四角で塗りつぶして、前の残り時間を消します。
つぎに、新しい残り時間をPRINT＃文によって表示します。

890行：残り時間が0でないときは、割り込み処理を終了して割り込みの発生したところへもどります。
900行からは、「ゲームオーバー」の処理になります。

900行：得点(スコア：変数SC)が最高点(ハイスコア：変数HSC)より大きいときは、HSCに新しい数を入れます。

910行・920行：ハイスコアを書き直します。

930行：スペースキーを押したときの割り込みを禁止します(何回も発射音が出るのを防ぐため)。また、表示色を白として、「ゲームオーバー」の表示をするサブルーチンへ移ります。

990行・1020行：PRINT＃文によって、「ゲームオーバー」を表示します。
930行からのサブルーチンとして実行されると、文字を白色とします。
950行からのサブルーチンとして実行されると、文字を黒色として表示します。つまり、前に書いた文字を消します。

940行：キー入力をチェックし、リターンキーの入力を待ちます。リターンキーが押されると950行へ移りゲームを再開します。

```
IF INKEY$ =CHR$(13)  THEN～
```

CHR$(13)：リターンキーのコントロールコード

INKEY$：キー入力された文字(何も入力しなければ、ヌルストリングとなります)。

950行：表示色を黒色として「ゲームオーバー」の表示をするサブルーチンへ移ります（つまり文字を消します)。
その後、表示色を白色とします。

960行・970行：ゲームを再開するため、スコアの表示を0とします。

980行：MFを0として、ミサイルを消します。
RETURN 470によって、タイマー割り込みの発生したところではなく470行へもどります。
これにより、時間、スコア、色などの初期設定がされます。

第2章のプログラムは以上です。ここまでに説明した命令・関数を参考にして、より楽しいプログラムを作ってください。

ちょっと一言

■割込み

ここで使った3つの割り込み以外にもいろいろな割り込み命令がありますが、いずれもその処理ルーチンを宣言した行からジャンプするのではなく、割込みの条件になったところからジャンプします。それに対してRETURN文は、何も指定しなければ割込みの発生したところへもどりますが、もどる行を指定することもできます。

第 3 章 文 法

（コマンド，ステートメント，関数の説明）

第 3 章はMSX BASICで使う命令や関数などをアルファベット順に説明しています。

用語 ───→

ABS

読み方 意味 ───→ absolute（アブソリュート：絶対の，絶対的な）　　関数

働き　数の絶対値を得ます。

書き方　ABS（〈数や式など〉）

例　A=ABS(−3)：PRINT　ABS（A∗B−2）

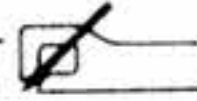
説明　（　）内の数値や数式の絶対値を計算します。関数のため、他の命令と組み合せて使います。また、（　）内の数値が単精度でも整数でもかまいませんが、得られる数値は倍精度です。

サンプルプログラム

数値（−5から5）を絶対値で表示します。

```
10 PRINT "A ﾊ -3ｶﾗ3    ABS(A)
20 FOR I=-3 TO 3
30 PRINT "    ";I,ABS(I)
40 NEXT I
```

```
A ﾊ -3ｶﾗ3    ABS(A)
 -3            3
 -2            2
 -1            1
  0            0
  1            1
  2            2
  3            3
```

言葉の説明 ───→ 用語　絶対値とは？

絶対値は、数の大きさだけを表わします。ですから数に−(マイナス)がついていてもいなくても同じ大きさになります。　例：ABS(−3)=ABS(3)

ASC

ASCII（アスキー）　　関数

働き　（　）内の文字のキャラクターコードを得ます。

書き方　ASC（〈文字〉）

例　B=ASC（"d"）

説明　文字（文字列のときは最初の文字）のキャラクターコードを得ます。他の命令と組み合せて使います。文字とキャラクターコードとの対応は、第5章の「キャラクターコード表」をご覧ください。

注意！　〈文字〉が""（ヌルストリング）のときには、Illegal function call エラーとなります。

サンプルプログラム

キー入力した文字のASCコードを表示します。

```
10 A$="":C$=""
20 INPUT "ﾓｼﾞﾊ ";A$
30 IF A$="" THEN 20
40 B$=LEFT$(A$,1):B=ASC(B$)
50 IF B<>1 THEN 80
60 B=ASC(MID$(A$,2,1))-64
70 B$=LEFT$(A$,2):C$="GRAPH"
80 PRINT B$;" ﾉASCｺｰﾄﾞﾊ ";C$;B;" ﾃﾞｽ"
90 PRINT:GOTO 10
```

```
run
ﾓｼﾞﾊ ? 5
5 ﾉASCｺｰﾄﾞﾊ  53  ﾃﾞｽ

ﾓｼﾞﾊ ? ｻ
ｻ ﾉASCｺｰﾄﾞﾊ  187  ﾃﾞｽ

ﾓｼﾞﾊ ? 木
木 ﾉASCｺｰﾄﾞﾊ GRAPH 4  ﾃﾞｽ

ﾓｼﾞﾊ ? ♣
♣ ﾉASCｺｰﾄﾞﾊ  130  ﾃﾞｽ

ﾓｼﾞﾊ ?
```

関係する用語 ───→ 参照　資料「キャラクターコード表」，CHR$(　)

第3章の見方

働　き **命令の機能、使い方を簡単に示します。**

書き方 **命令の書き方を示します。実際に入力するときは、次のきまりに従ってください。**

i ）命令を入力するときは、アルファベットの大文字でも小文字でもかまいません。ただし、カセットなどにセーブ、ロードするときのファイル名は、大文字と小文字を区別してください。

ii ）カギカッコ〈　〉で囲まれた項目は、あなたが指定します。

iii ）角カッコ〔　〕で囲まれた項目は、省略することができます。

例．COLOR〔〈前景色〉〕〔,〈背景色〉〕〔,〈周辺色〉〕

⬆

〈背景色〉、〈周辺色〉を指定しないときは省略できる。

ただし、角カッコに続くパラメータのいずれかを指定するときは、それ以前の角カッコに含まれるカンマ（，）などの区切り記号は省略できません。

例．COLOR15，，7

iv ）省略記号●●●の続く項目は、1行の許す範囲内で何回も繰り返すことができます。

v ）｛　｝の記号で書かれたものは、上下どちらの書き方を使ってもかまいません。

vi ）上記以外の記号は、文法的に必要な記号ですから、示された位置に正しく入力してください。

vii ）書き方の例として、実際の文を書いてあるものについては、右側にその実行結果も示してあります。

説　明 命令の使い方や、詳しい機能、注意点をまとめています。

サンプルプログラム いくつかの命令を使った簡単なプログラム例です。必要なものに対してはプログラムの横にその実行結果も示してあります。

参　照 関係のある命令や資料などを示しています。

命令、関数などの説明は、MSX、MSX2を区別しないで説明していますが、区別する必要がある場合は、MSX2専用の部分にMSX2と示します。MSX2については第5章でも説明しています。
また、MSX2+のパソコンではMSX2+の拡張機能を使わないとき、MSX、MSX2の命令や関数がそのまま使えます。MSX2+の拡張機能を使うときの注意点については、MSX2+パソコンに付属の「拡張BASIC説明書」をご覧ください。

注意： この章では、MSX BASICで使う命令のほかにDisk BASICの命令を参考に記載しています。
MSX Disk BASICは、アスキーの商標です。

ABS

absolute（アブソリュート：絶対の，絶対的な）　関数

働き 数の絶対値を得ます。

書き方 ABS（〈数や式など〉）

例　A=ABS(－3)：PRINT　ABS(A－2)　　実行結果……1

説明 （　）内の数値や数式の絶対値を計算します。関数のため、他の命令と組み合せて使います。また、（　）内の数値が単精度でも整数でもかまいませんが、得られる数値は倍精度です。

サンプルプログラム

数値（－3から3）を絶対値で表示します。

```
10 PRINT "A ﾊ -3ｶﾗ3    ABS(A)
20 FOR I=-3 TO 3
30 PRINT "    ";I,ABS(I)
40 NEXT I
```

```
A ﾊ -3ｶﾗ3    ABS(A)
     -3          3
     -2          2
     -1          1
      0          0
      1          1
      2          2
      3          3
```

用語 絶対値とは?

絶対値は、数の大きさだけを表わします。ですから数に－(マイナス)がついていてもいなくても同じ大きさになります。　例：ABS(－3)＝ABS(3)

ASC

ASCII（アスキー）　関数

働き （　）内の文字のキャラクターコードを得ます。

書き方 ASC（〈文字〉）

例　B=ASC("d")：PRINT B　　実行結果……100

説明 文字（文字列のときは最初の文字）のキャラクターコードを得ます。他の命令と組み合せて使います。文字とキャラクターコードとの対応は、第5章の「キャラクターコード表」をご覧ください。

注意！ 〈文字〉が "" (ヌルストリング) のときには、Illegal function call エラーとなります。

サンプルプログラム

キー入力した文字のASCコードを表示します。

```
10 A$="":C$=""
20 INPUT "ﾓｼﾞﾊ ";A$
30 IF A$="" THEN 20
40 B$=LEFT$(A$,1):B=ASC(B$)
50 IF B<>1 THEN 80
60 B=ASC(MID$(A$,2,1))-64
70 B$=LEFT$(A$,2):C$="GRAPH"
80 PRINT B$;" ﾉASCｺｰﾄﾞﾊ ";C$;B;" ﾃﾞｽ"
90 PRINT:GOTO 10
```

```
run
ﾓｼﾞﾊ ? 5
5 ﾉASCｺｰﾄﾞﾊ  53  ﾃﾞｽ

ﾓｼﾞﾊ ? ｻ
ｻ ﾉASCｺｰﾄﾞﾊ  187  ﾃﾞｽ

ﾓｼﾞﾊ ? 木
木 ﾉASCｺｰﾄﾞﾊ GRAPH 4  ﾃﾞｽ

ﾓｼﾞﾊ ? ♣
♣ ﾉASCｺｰﾄﾞﾊ  130  ﾃﾞｽ

ﾓｼﾞﾊ ?
```

参照 資料「キャラクターコード表」，CHR$(　)

ATN

arc tangent (アークタンジェント：逆正接)　関数

働き　逆正接（アークタンジェント）を得ます。

書き方　ATN（〈数〉）

例　PRINT ATN(1)　実行結果……0.78539816339745

説明　〈数〉の逆正接（アークタンジェント：三角関数の1つ）を与えます。（　）内の〈数〉は整数でも単精度でもかまいませんが、得られる値は $-\pi/2$ から $\pi/2$ までの倍精度です。また、値の単位はラジアンです（π=3.1416）。

AUTO

auto (オート：自動的に)　コマンド

働き　プログラムを入力するときに、行番号を自動的に作ります。

書き方　AUTO〔〈開始行番号〉〕〔,〈増分〉〕

例　AUTO 100, 20

説明　プログラムを入力するときに便利なように、行番号を自動的に作り画面に表示します。
このコマンドを実行すると、〈開始行番号〉を画面に表示してプログラムの入力を待ちます。次にリターンキーを押すと〈開始行番号〉に〈増分〉を加えた行番号が表示されます。以後、リターンキーを押すごとに〈増分〉ずつ加えた行番号が自動的に発生します。

〈開始行番号〉のみを省略すると0が〈開始行番号〉となり、〈増分〉のみを省略すると10が〈増分〉となります。また、〈開始行番号〉と〈増分〉を両方とも省略するとそれぞれ10が自動的に指定されます。

[CTRL]+[C]または[CTRL]+[STOP]を押すと、行番号の自動発生を終了してBASIC のコマンドレベルに戻ります。このとき、最後に表示された行番号の行は入力されません。

注意！　すでに入力されている行番号と同じ行番号を発生したとき、行番号のすぐ後に「*」が表示されて、キー入力するとプログラムが変更されることを注意します。このときリターンキーだけを押すとその行のプログラムはそのまま残ります。

BASE

base（ベース：基本）　システム変数

働　き　VDPレジスタの各テーブルの先頭アドレスを得ます。

書き方　BASE（〈数〉）

例　A＝BASE（10）

説　明　画面表示を行なうVDP（ビデオディスプレイプロセッサー）レジスタ内の各テーブルの先頭アドレスを得ます。指定できる数値は0～19（MSX2では0～44）です。
このシステム変数に値を代入することはできません。

参　考　VDP, VPEEK

B

BEEP

beep（ビープ：ビーと鳴らす）　ステートメント

働　き　スピーカーを鳴らします。

書き方　BEEP

説　明　スピーカーを約0.04秒鳴らします。
BEEPはPRINT CHR$(7)とPRINT命令を使っても同じ結果が得られます。

サンプルプログラム

1秒ごとにスピーカーを鳴らします。

```
10 LOCATE 2,3:PRINT "TIME "
20 ON INTERVAL=60 GOSUB 70
30 TIME=0:INTERVAL ON
40 T=TIME/60
50 LOCATE 7,3:PRINT USING"###.#秒";T
60 GOTO 40
70 BEEP:RETURN
```

```
run

 TIME 120.5秒
```

参　照　資料「コントロールコード表」，SET BEEP（MSX2）

BIN $

binary $(バイナリー　ドル：2進数の文字列)　関数

働　き 数値を2進数の文字列に変え、その結果を得ます。

書き方 BIN　$(〈数値〉)

例　PRINT "&B"+BIN$(8)　　実行結果……&B1000

説　明 〈数値〉を2進数で表わす文字列に変えます。〈数値〉の値の範囲は－32768から32768までの整数です。変換結果の文字列は、ゼロサプレス（先頭の不要な0を取り除く）されています。

サンプルプログラム

10進数を2進数、8進数、16進数で表示します。

```
10 PRINT " A  BIN$(A) OCT$(A) HEX$(A)"
20 PRINT "————————————————————————————— "
30 FOR A=1 TO 16
40 A$=RIGHT$("      "+BIN$(A),6)
50 B$=RIGHT$("      "+OCT$(A),5)
60 C$=RIGHT$("      "+HEX$(A),5)
70 PRINT USING"## 1@   1@   1@";A;A$;B$;C$
80 NEXT
```

A	BIN$(A)	OCT$(A)	HEX$(A)
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

参　考 OCT$, HEX$

BLOAD

bload(ビーロード)　コマンド

働　き 機械語のプログラムをロードします。

書き方 BLOAD "〈ファイルスペック〉" ｜〔,R〕〔,S〕｜〔,〈オフセット〉〕

例　BLOAD "CAS：LPN",R

説　明 〈ファイルスペック〉で指定した機械語のプログラムを読み込み（ロード）します。
Rをつけると、プログラムのロードが終った後、BSAVE のとき指定しておいたプログラムの実行開始番地からただちに実行します。Sは、画像データのロードのときに指定します。

〈オフセット〉を設定すると、BSAVE で指定されていたアドレス(開始アドレス)に〈オフセット〉の値だけ加えてロードされます。このため、〈オフセット〉を設定するプログラムは、アドレスを変えても実行するもの（リロケータブル）でなければなりません。

参　考 BSAVE

ちょっと一言

〈ファイルスペック〉は使用する周辺機器により変更できます。

BLOAD "CAS:LPN",R …… カセットから読み込みます
BLOAD "A:LPN",R ………… ディクスから読み込みます
BLOAD "LPN",R ………… ディスクを接続しているときは、ディスクから読み込みます
ディスクを接続していないときは、カセットから読み込みます。
BLOAD "CAS:",R ………… カセットから読み込みます。（ファイル名の指定はない）

BSAVE

bsave（ビーセーブ）　コマンド/Disk BASIC

働　き　機械語プログラムをセーブします。

書き方　BSAVE "〈ファイルスペック〉",〈開始アドレス〉,〈終了アドレス〉〔,〈実行開始アドレス〉〕〔,S〕

例　BSAVE "CAS：LPN", &H8010, &HD000, &H8010

説　明　パソコンのメモリにある機械語プログラムを〈ファイルスペック〉で指定したファイルにセーブします。また、VRAMにある画像データもセーブできます。
メモリのどこからどこまでをセーブするかを〈開始アドレス〉と〈終了アドレス〉で指定します。上の例では、 &H8010から&HD000までのメモリの内容をLPNという名前の機械語プログラムとしてカセットテープにセーブします。
〈実行開始アドレス〉は、BLOAD 命令でRをつけたとき、ロードした後に機械語プログラムを実行するときのアドレスです。また、〈実行開始アドレス〉を省略すると、〈開始アドレス〉が〈実行開始アドレス〉とされます。
なお、プログラムはバイナリ形式で記録されます。
Sをつけたとき、VRAMにある画像データをディスクにセーブできます。

参　照　BLOAD

ちょっと一言

〈ファイルスペック〉中の〈デバイス名〉は、周辺機器により変更します。
"CAS："………………カセットレコーダにセーブします。
"A：" または "B：" …ディスクにセーブします。
〈デバイス名〉を省略すると、ディスクを接続しているときはディスクに、ディスクを接続していないときはカセットレコーダにセーブされます。

B

CALL

call（コール：呼ぶ）　ステートメント

働　き　用意されている拡張ステートメントを呼び出します。

書き方　CALL〈拡張ステートメント名〉〔(〈式〉〔,〈数〉〕,…)〕

例　CALL FORMAT
_FORMAT

説　明　拡張ROMカートリッジなどで用意された拡張ステートメントを呼び出します。"CALL"の代りに アンダーバー（_）を使うこともできます。

注意！　この命令の使い方は、各ROMカートリッジの説明書をご覧ください。

CALL MEMINI/MFILES/MKILL/MNAME

MSX2

MSX2では、メインRAMの一部をフロッピーディスクのようにプログラムなどの保管場所として使える機能があります（RAMディスクといいます）。

memini（メモリ・イニシャライズ：RAMディスクを初期化する） MSX2

働き RAMディスクとして使うメインRAMの上限を設定し、ファイルを初期化します。

書き方 CALL MEMINI〔(〈RAMディスクの上限値〉)〕

例 CALL MEMINI（&H 3 FFF） 実行結果……15616 bytes allocatedと表示される。

説明 RAMディスクとして使うメモリの範囲を指定し、RAMディスクの設定を初期化（ファイル、メモリ情報をクリア）します。指定できる上限値は、&H3FF～&H7FFFです。値を省略すると&H7FFFが指定されます。たとえば、上限値に&H3FFを指定すると使用できるメモリは、&H0～&H3FFまでの範囲です。&H3FE以下を指定すると No RAM disk の表示がでてRAMディスクの機能は解除されます。

RAMディスクを使用するためには、必ず他の入出力命令を実行する前にMEMINI命令でRAMディスクの使用を宣言しなければなりません。

注意！ 下限値を設定することはできません。また、設定値は&H 3 FFから&H100ステップごとに有効です（例. &H 4 FF～&H5FEは&H 4 FFと同じ）。

mfiles（メモリ・ファイル：RAMディスク内のファイル一覧） MSX2

働き RAMディスク内のファイル名とRAMディスクの残り容量を表示します。

書き方 CALL MFILES

mkill（メモリ・キル：RAMディスク内のファイル削除） MSX2

働き RAMディスク内のファイルを削除します。

書き方 CALL MKILL（"〈ファイル名〉"）

例 CALL MKILL（"MSX"） 実行結果……MAXというファイルを削除する。

注意！ OPEN命令で開いたファイルはCLOSE命令で閉じるまで削除できません。

mname（メモリ・ネーム：RAMディスク内のファイル名変更） MSX2

働き RAMディスク内のファイルの名前を付け替えます。

書き方 CALL MNAME（"〈現在のファイル名〉" AS "〈新ファイル名〉"）

例 CALL MNAME（"MSX2" AS "MSX"）

参照 第1章「ファイルについて」、KILL，NAME，OPEN

ちょっと一言

● 次ぎの命令がDisk BASICと同様に使えます。
SAVE，LOAD，RUN，MERGE，OPEN，CLOSE，PRINT #，PRINT # USING，INPUT#，LINE INPUT#，INPUT$，EOF，LOC，LOF

CDBL

c-double（シーダブル：2倍の） 関数

働　き 数値を倍精度実数に変え、その値を得る。

書き方 CDBL（〈数〉）

例　PRINT CDBL(1/6)　　実行結果…….16666666666666

説　明 〈数〉の値を倍精度実数に変えます。ただし、型変換が行なわれるだけで有効桁数は変わりません。

例）A1＝3.14159 のとき、B＝CDBL（A1）としても
B＝3.14159 です。（B＝3.1415900000000 です）

ちょっと一言 実際には、型を指定しないで計算をしたり、倍精度実数型の変数に代入しただけで倍精度実数に変えられます。

例）PRINT　A!＊B!
A＝B!
（倍精度に変えられている。）

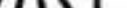
CHR$

character $（キャラクタードル：文字，記号） 関数

働　き キャラクターコードに対応する文字を得る

書き方 CHR$（〈数〉）

例　PRINT CHR$(33)　　実行結果……！

説　明 〈数〉のキャラクターコードの文字を得ます。
〈数〉の値が32〜255のとき、そのキャラクターコードの文字を得ます。
〈数〉の値がコントロールコード（0〜31）のとき、そのコントロールコードの機能が実行されます。
〈数〉が 0〜255 の範囲でないときは、“Illegal function call” エラーとなります。

サンプルプログラム

すべてのキャラクターを表示します。

```
10 FOR K=32 TO 255
20 PRINT CHR$(K);" ";
30 NEXT K
40 FOR M=64 TO 95
50 PRINT CHR$(1)+CHR$(M);" ";
60 NEXT M
```

```
  !  "  #  $  %  &  '  (  )  *  +  ,  -  .  /  0  1  2  3
4 5  6  7  8  9  :  ;  <  =  >  ?  @  A  B  C  D  E  F  G
H I  J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X  Y  Z  [
¥ ]  ^  _  `  a  b  c  d  e  f  g  h  i  j  k  l  m  n  o
p q  r  s  t  u  v  w  x  y  z  {  |  }  ‾     ♠  ♥  ♣  ♦  ○
 ● を ぁ ぃ ぅ ぇ ぉ ゃ ゅ ょ っ    あ い う え お か き く
 け こ さ し す せ そ    。 「 」 、 ・ ヲ ァ ィ ゥ ェ ォ ャ
 ュ ョ ッ ー ア イ ウ エ オ カ キ ク ケ コ サ シ ス セ ソ タ
 チ ツ テ ト ナ ニ ヌ ネ ノ ハ ヒ フ ヘ ホ マ ミ ム メ モ ヤ
 ユ ヨ ラ リ ル レ ロ ワ ン ゛ ゜ た ち つ て と な に ぬ ね
 の は ひ ふ へ ほ ま み む め も や ゆ よ ら り る れ ろ わ
 ん    ■    月 火 水 木 金 土 日 年 円 時 分 秒 百 千 万 π
```

参　照 ASC、「キャラクターコード表」

CINT

c-integer（シーインテジャー：整数）　関数

働き　**数値を整数に変え、その値を得る。**

書き方　CINT（〈数〉）

例　A%=CINT（B!/3）

説明　〈数〉の小数点以下を切り捨てて、整数にします。結果が －32768～32767 の範囲でないときは、"Overflow" エラーとなります。

ちょっと一言　整数型の変数に値を代入しただけでも、整数に変えられます。
　例）B#=11.9：A%=B#/3　←A%は整数の3です。

CIRCLE

circle（サークル：円）　ステートメント

働き 円、弧を描きます。

書き方 CIRCLE（〈X座標〉,〈Y座標〉),〈半径〉〔,〈色〉〕〔,〈開始角度〉〕〔,〈終了角度〉〕〔,〈比率〉〕

例　CIRCLE (117, 100), 50, 15, 0, 2

説明 〈X座標〉、〈Y座標〉を中心とした〈半径〉の大きさの円を描きます。
円の色は〈色〉のカラーコードで指定され、〈色〉を省略すると、COLOR命令で指定している前景色(ぜんけいしょく)で描かれます。

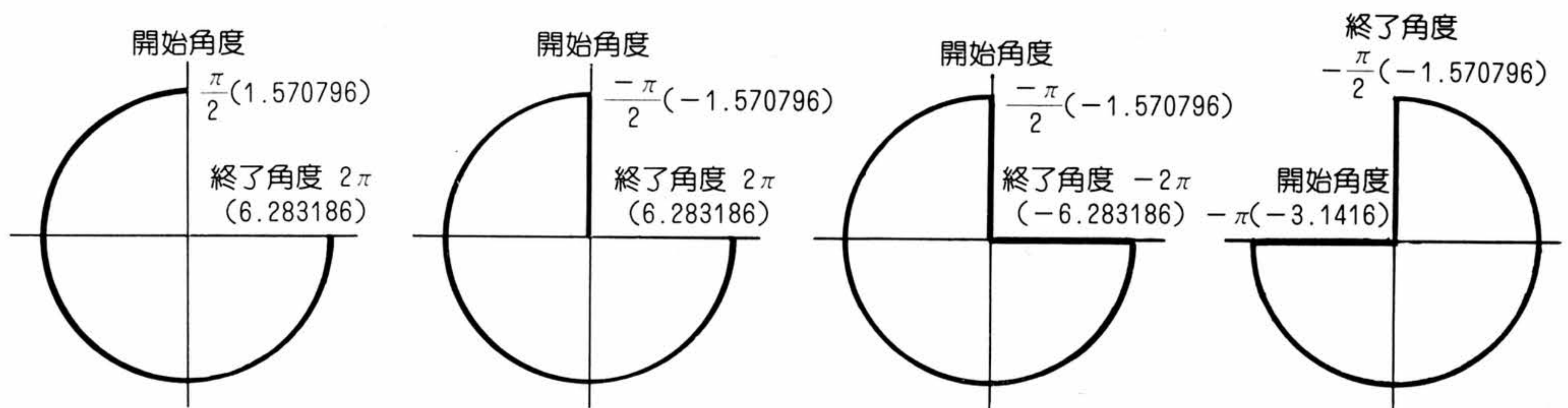
〈開始角度〉、〈終了角度〉を指定すると、指定した角度の範囲の弧が描かれます。角度を省略すると〈開始角度〉に0が、〈終了角度〉に 6.283186 (2π) が使われます。
角度が負の値のとき、その絶対値の角度を使いますが、そのとき中心から半径が描かれるので扇形(おうぎ)を描くことができます。

角度が−2π〜2πの範囲にないときや、グラフィックモード以外でCIRCLE命令を使うと "Illegal function call" エラーとなります。πは、3.141593です。

開始角度 $\frac{\pi}{2}$(1.570796)　終了角度 2π (6.283186)

開始角度 $\frac{-\pi}{2}$(−1.570796)　終了角度 2π (6.283186)

開始角度 $\frac{-\pi}{2}$(−1.570796)　終了角度 −2π (−6.283186)

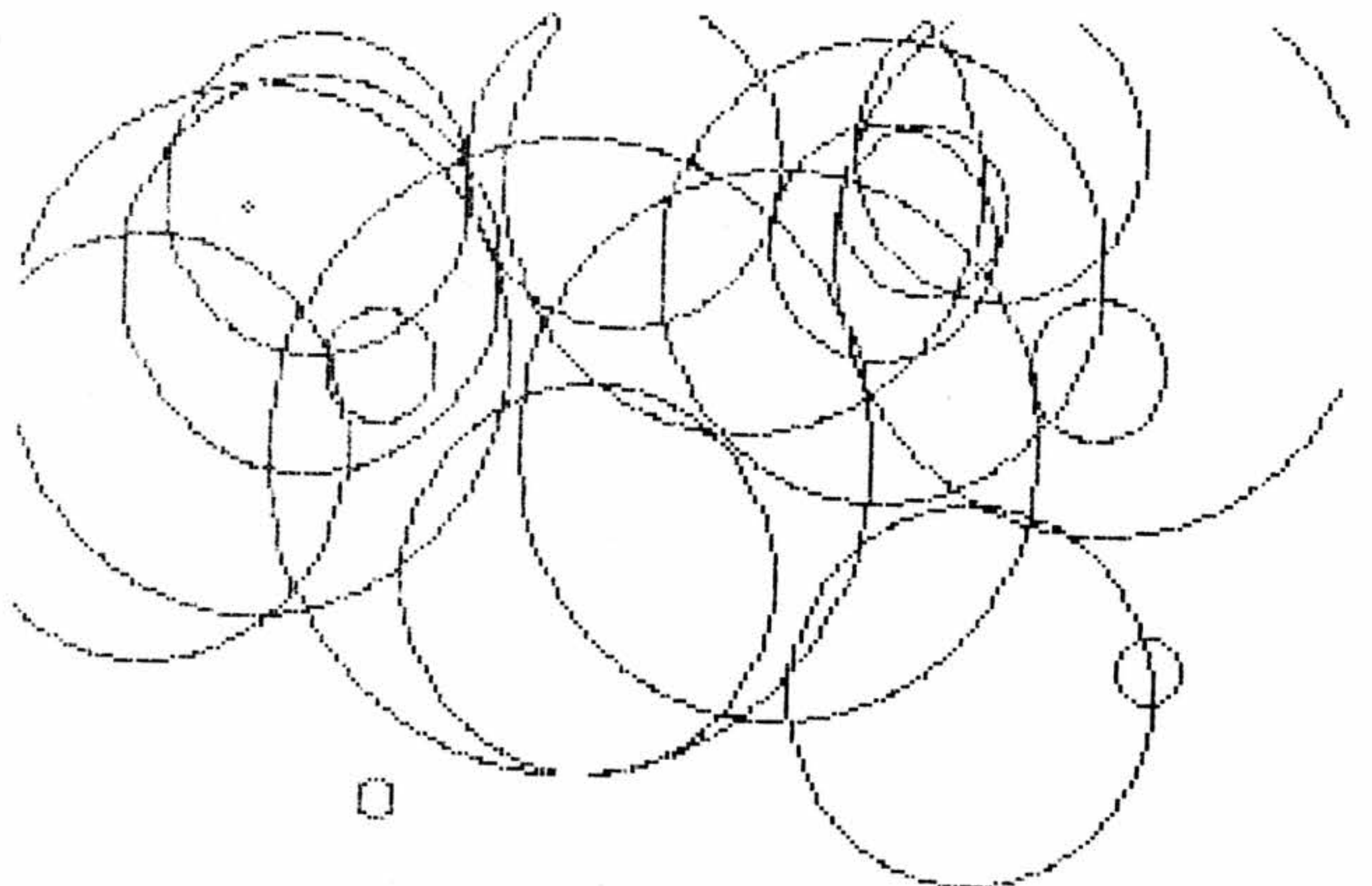
終了角度 $-\frac{\pi}{2}$(−1.570796)　開始角度 −π(−3.1416)

〈比率〉は円の縦横の割合です。1より小さいとき横長の円になり1より大きいときは縦長の円となります。〈比率〉が1より小さいとき、〈半径〉は横（水平）方向の大きさを表わします。〈比率〉が小さい大きいときは縦方向の大きさを表わします。

中心の位置は、STEPを付けてLP（最終参照点）からの相対座標で指定することもできます。　例）CIRCLE STEP (10, 20), 50, 15,,, 1.15

また、CIRCLE 命令を実行するとLPは円の中心（〈X座標〉,〈Y座標〉）になります。

サンプルプログラム

ランダムに円を描きます。

```
10 '** CIRCLE **
20 COLOR 15,15,7:SCREEN 2
30 FOR K=1 TO 20
40 X=RND(1)*200+20
50 Y=RND(1)*150+20
60 R=RND(1)*70
70 C=RND(1)*14+1
90 CIRCLE(X,Y),R,C,,,1.15
100 NEXT K
110 GOTO 110
```

CLEAR

clear（クリアー：明確な，はっきりした）　ステートメント

働　き　変数を初期化し、使用するメモリの大きさを設定します。

書き方　CLEAR〔〈文字列の大きさ〉〔,〈メモリの上限番地〉〕〕

例　CLEAR100，&HD000

説　明　ベーシックで使用するメモリの上限番地を設定します。このとき、すべての変数を0に、文字変数を""（ヌルストリング）にします。また、オープンしているファイルがあれば、すべてクローズされます。

〈文字列の大きさ〉は、文字変数の内容をしまっておくメモリの大きさ（文字列領域）を示します。電源を「入」にしたときには〈文字列の大きさ〉は200バイトに設定されています。（1バイトは1文字分のメモリです）

〈メモリの上限番地〉を設定すると、その番地の直前までをベーシックで使うメモリ（プログラムや変数の記憶に使います）としますので、その番地以降に機械語プログラムなどを置くとベーシックで破壊されません。

〈メモリの上限番地〉を設定するときは、〈文字列の大きさ〉を省略することはできません。

注意！　〈メモリの上限番地〉で設定できる番地は、
16KB RAMのパソコンでは、&HC31F～&HF380
32KB RAMのパソコンでは、&H831F～&HF380
です。（64KBRAMのパソコンでも32KB RAMのパソコンと同じです）。

また、CLEAR命令によりDEF文（DEFFN，DEFUSRなど）で定義した条件はすべて無効となりますので、DEF文より先にCLEAR命令を実行します。

サンプルプログラム　変数Aに代入された数値5がCLEAR文で0になります。

```
10 A=5           run
20 PRINT A        5
30 CLEAR          0
40 PRINT A       Ok
```

参　照　FRE，資料「メモリマップ」

ちょっと一言　CLEAR命令に関係して、次のようなエラーメッセージが表示されることがあります。

- **File not OPEN**
  CLEAR命令で、オープンしていたファイルがクローズしたとき。
- **Illegal function call**
  〈文字列の大きさ〉や〈メモリの上限番地〉が実際には取れないような値のとき。
- **Out of memory**
  ベーシックのプログラムが長くなったり、変数が多いなどの原因でメモリがたりなくなったとき。
- **Out of string space**
  文字列や文字変数が多かったり、長いなどの原因で〈文字列の大きさ〉で設定したメモリがたりなくなったとき。

CLOAD／CLOAD?

c-load（シーロード：仕入れる）
c-load?（シーロード・ベリファイ）　　コマンド

働　き　カセットテープからメモリへプログラムを読み込みます。
メモリのプログラムとカセットテープのプログラムを比較します。

書き方　CLOAD〔"〈ファイル名〉"〕
CLOAD?〔"〈ファイル名〉"〕

例　CLOAD "〈DEMO〉"
CLOAD?"〈DEMO〉"

説　明　CLOADはカセットテープから〈ファイル名〉で指定したプログラムを探し出してロード(読み込み)します。目的のプログラムを見つけるとFound：〈ファイル名〉と表示され、プログラムがロードされます。探しているときに別のプログラムを見つけるとSkip：〈ファイル名〉と表示します。

CLOAD?は、メモリにあるプログラムと〈ファイル名〉で指定されたテープのプログラムを比較し、内容が同じであればOKと表示し、もしそうでなければVerify errorと表示されます。(Verify：ベリファイ…確かめる)
この比較は、メモリのプログラムが正しくテープにセーブされたか確かめるもので、CSAVEの直後に使います。

〈ファイル名〉は6文字以内で指定し、〈ファイル名〉を省略するとカセットテープから最初に読み込まれたプログラムがロードされます。

注意！　CLOADを実行し、Found：が表示されると、メモリにあったプログラムは消えてしまいます。プログラムの合成には、MERGEを使います。また、CLORD，CLORD?命令で、BSAVE,SAVE命令によってセーブされたプログラムをロードで比較することはできません。

参　照　CSAVE，LOAD，MERGE

C

CLOSE

close（クローズ：閉じる）　コマンド

働　き　入出力ファイルを閉じます。

書き方　CLOSE〔〔#〕〈ファイル番号〉〔,〔#〕〈ファイル番号〉〕・・・〕

例　CLOSE #1

説　明　指定した〈ファイル番号〉に対応する入出力ファイルを閉じます。〈ファイル番号〉を指定しないでCLOSE文を実行すると、開かれているファイルをすべて閉じます。
閉じたファイルに対しては、再びOPEN文で開くまで入出力はできません。

CLOSE文はファイルが出力用にオープンされていた場合には、バッファに残っていたデータをクリアしますので、ファイルの入出力を正しく終了するためには、必ずCLOSE文を実行してください。
また、END命令やNEW命令を実行すると、自動的にすべてのファイルが閉じられますがSTOP命令ではファイルは閉じません。

参　照　「ファイル」，OPEN，PRINT#，INPUT#，

CLS

clean screen（クリーン・スクリーン）　ステートメント

働　き　画面をクリアする

書き方　CLS

説　明　画面上の文字や絵をすべて消去します。
テキストの画面のとき、KEY ONの状態であれば画面下のファンクションキー表示だけは残り、カーソルは左上隅へ移動します。また、スプライトの表示も消えません。

グラフィック画面のとき、画面をクリアしてもLP(最終参照点)は変化しません。

COLOR

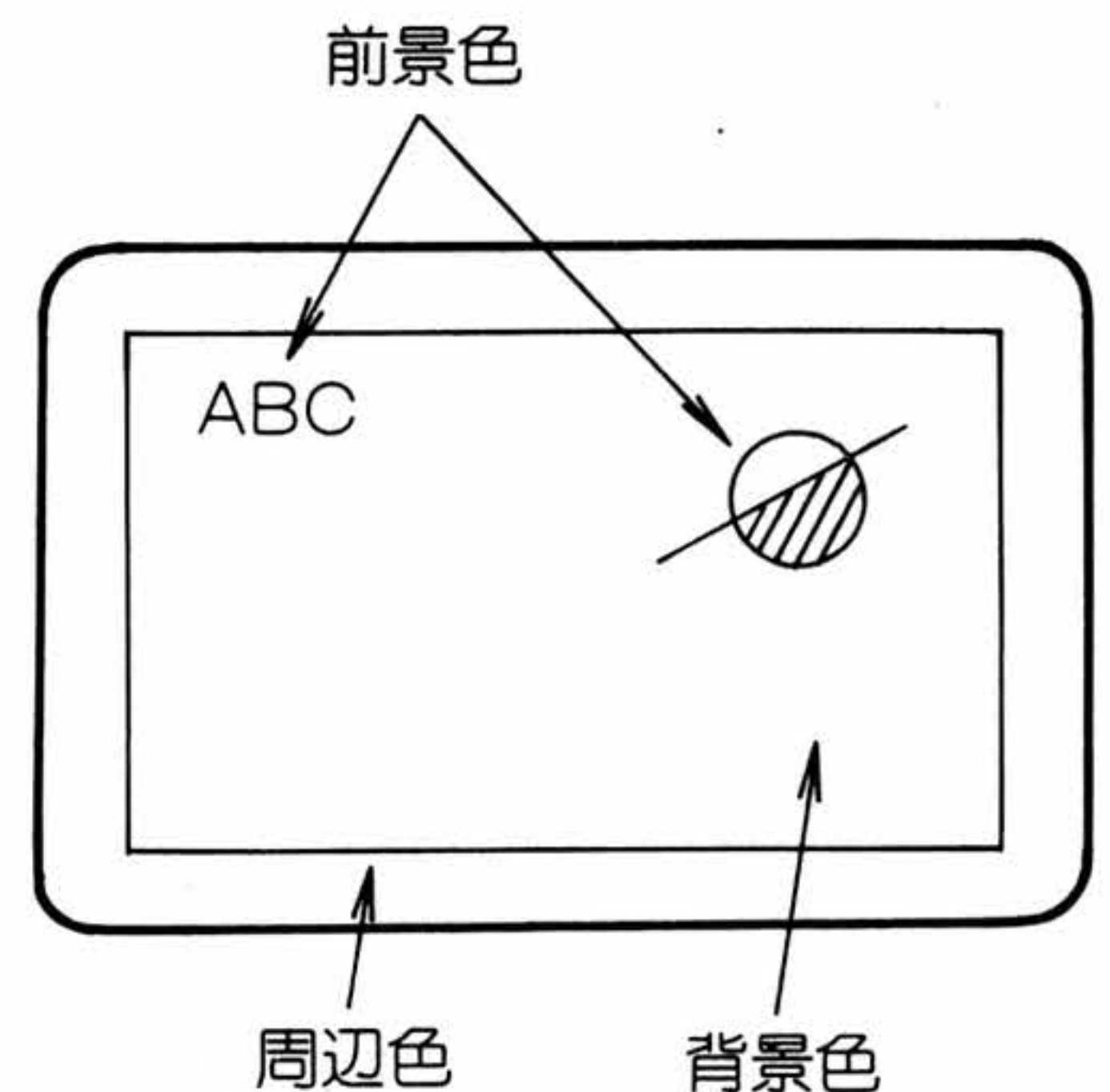
color（カラー：色）　ステートメント

働　き　画面の色を指定します。

書き方　COLOR〔〈前景色〉〕〔,〈背景色〉〕〔,〈周辺色〉〕

例　COLOR 15, 4, 1　　実行結果……前景色が白に、背景色が暗い青に、周辺色が黒になります。

説　明　テキスト画面の文字の色を変えたり、グラフィック画面の背景色、周辺色を指定します。色は次のようなカラーコードで指定します。

0――透明（周辺色と同じ）	8――赤
1――黒	9――明るい赤
2――緑	10――黄
3――明るい緑	11――明るい黄
4――暗い青	12――暗い緑
5――明るい青	13――紫
6――暗い赤	14――灰
7――水色	15――白

〈前景色〉はテキスト画面の文字やグラフィック画面に描く点や線の色です。PSET, LINEなどの命令で色を指定しないときは、この〈前景色〉が使われます。
前景色だけを変えるときは、次のように指定します。

COLOR 9

〈背景色〉はテキスト画面やグラフィック画面の地の色です。テキスト画面では命令を実行するとすぐに色が変りますが、グラフィック画面では、SCREEN命令かCLS命令を実行した後で背景色が変えられます。
背景色だけを変えるときは次のように指定します。

COLOR ,1

〈周辺色〉は、ベーシックで使う領域の外の色です。
周辺色だけを変えるときは、次のようにします。

COLOR ,,10

指定する色のカラーコードを省略すると、そのままの状態を保ちますが、3つの指定全部を省略することはできません。
電源を「入」にしたときは、15, 4, 7が設定されます。（MSX2では、SET SCREEN命令によって色を設定することができます。）

COLOR

MSX2

MSX ではカラーコードの色はあらかじめ設定された16色ですが、MSX2 では16色の色あいを自由に変えることができます（カラーパレット機能といいます）。

color＝

MSX2

働　き カラーコードの色の設定を変えます。

書き方 COLOR＝（〈カラーコード〉,〈赤レベル〉,〈緑レベル〉,〈青レベル〉）

例 COLOR＝（15，７，６，５）　実行結果…カラーコード15に肌色を設定

説　明 MSX2 では、SCREEN 8 以外の画面の色をカラーコードで指定します。指定できるカラーコードはSCREEN 6 では 0 ～ 3 、他の画面では 0 ～15です。各カラーコードには下の表の色が初期状態として設定されていますが、COLOR＝命令を使って色を変えることができます。これによって512色の中から16色（または 4 色）が自由に選べます。
カラーコードに設定できる明るさのレベルは各 3 ビットで、指定できる値はいずれも 0 ～ 7 です。

上の例では、カラーコード15の色を、７，６，５として肌色に設定しています。

７	６	５
赤の明るさ	緑の明るさ	青の明るさ
（&B111）	（&B110）	（&B101）

サンプルプログラム

```
10 COLOR 15,4,7
20 COLOR=(15,7,0,0)
30 COLOR=(4,0,7,0)
40 COLOR=(7,0,0,7)
```

←前景色を赤に、背景色を緑に、周辺色を青に設定します。

color＝new

MSX2

働　き カラーコードの色の設定を初期状態にします。

書き方 COLOR〔＝NEW〕

例 COLOR

説　明 カラーコードの色の設定を、電源を〝入〟にしたときと同じ初期状態（下表）にします。

カラーコード	色	赤レベル	緑レベル	青レベル	カラーコード	色	赤レベル	緑レベル	青レベル
0	透明	0	0	0	8	赤	7	1	1
1	黒	0	0	0	9	明るい赤	7	3	3
2	緑	1	6	1	10	暗い黄	6	6	1
3	明るい緑	3	7	3	11	明るい黄	6	6	4
4	暗い青	1	1	7	12	暗い緑	1	4	1
5	明るい青	2	3	7	13	紫	6	2	5
6	暗い赤	5	1	1	14	灰	5	5	5
7	水色	2	6	7	15	白	7	7	7

このほかに、SCREEN命令を実行したとき、SCREEN 0 で40文字以下だった表示をWIDTH文で41以上に設定したとき、また、逆に41以上から40以下にしたときもカラーコードの設定が初期化されます。

color＝restore

MSX2

働　き　ビデオRAM内のカラールックアップテーブルに従って、カラーコードの色を設定します。

書き方　COLOR＝RESTORE

説　明　ビデオRAM内のカラーコードの色情報（カラールックアップテーブルのデータ）の内容に従って、画面表示用LSI（VDP）のカラーコードの色情報を書き変えます。この命令は主に、Disk BASICのBSAVE命令で画面データをフロッピーにセーブし、再びBLOAD命令で画面データをロードしたときに使います。
カラーコードの色情報は、ビデオRAMとVDPに記憶されていますが、画面を直接コントロールしているVDPの色情報はBLOADしただけでは書き変わりません。このため、セーブしたときと異なった色で表示されることがあります。COLOR＝RESTOREを実行すると、ビデオRAM内の色情報に従ってVDPの色を設定し直すので、セーブしたときと同じ色になります。（セーブするとき、ビデオRAMの色情報もセーブしておく必要があります。）

```
10 COLOR=(15,0,7,0)            }← 文字を緑色にして、MSXというファイル名でセーブします。
20 BSAVE"MSX",0,&H4000,S
30 COLOR                        }← その後、カラーを初期化し、画面を消します。
40 CLS
50 BLOAD"MSX",S                 }← 再び画面をロードすると異なった色で表示されます。
60 FOR K=0 TO 500:NEXT K
70 COLOR=RESTORE                ← COLOR=RESTORE文で色が再設定されます。
```


ちょっと一言

■カラーコードの色レベルについて

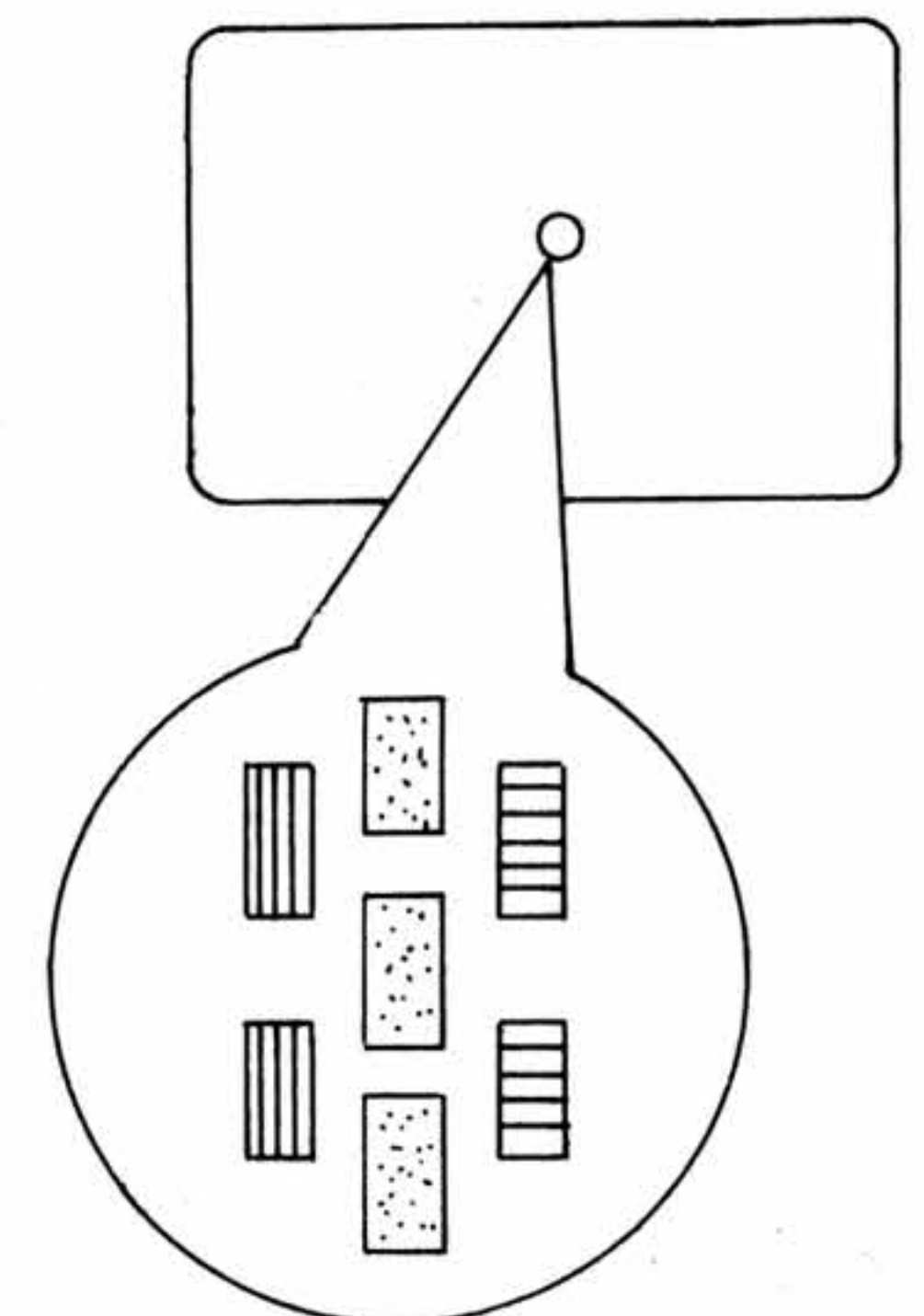
カラーテレビの画面は、赤、緑、青の点の集まりで作られています。そしてこの一つ一つの点の明るさを変えることによって何種類もの色（色あい）が表示されています。カラーコードの色レベルは、この個々の点の明るさを表わし、0～7の8段階に変えることができます。（0が暗く、7が明るい）。
赤、緑、青の3色の組合せで色は512色（8×8×8）となります。

赤、緑、青の点の集まりで色を作ります。

■カラーパレットとは?

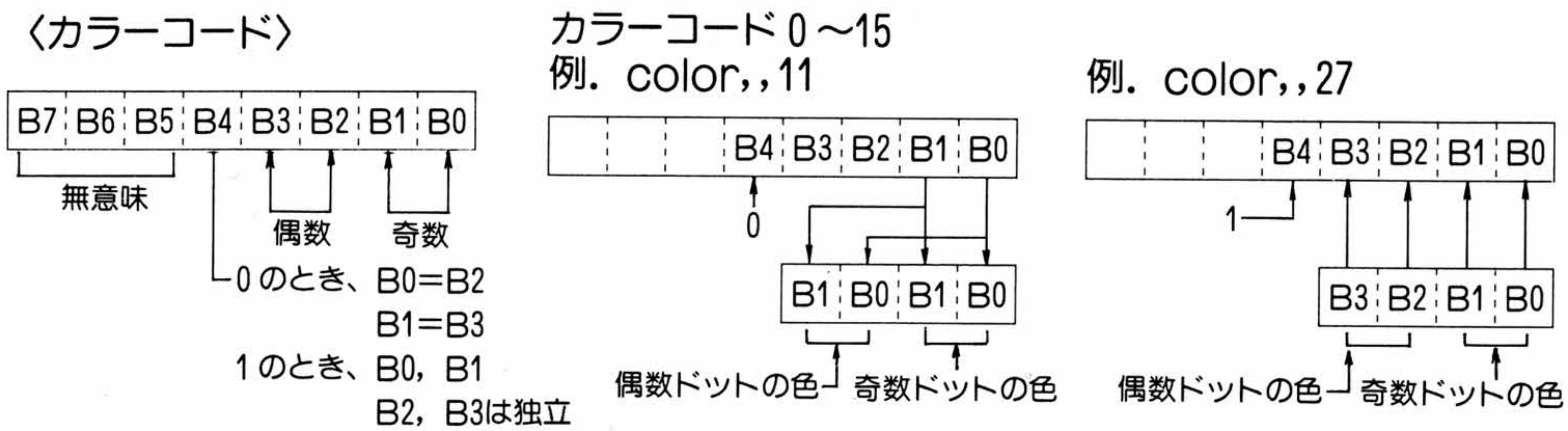
MSX のCOLOR命令で作るカラーコードの色は、あらかじめ決められていて変えることができません。しかし、MSX2ではカラーコードの色を変えることができるので、MSX と MSX2 のCOLOR命令を区別するため、MSX2のカラーコードをカラーパレットと呼ぶこともあります。

COLOR

MSX2

SCREEN 6の色

SCREEN 6では使用できるカラーコードは0〜3です。このため他の画面(例えばSCREEN 7)で使ったカラーコード0〜15は、SCREEN 6では次の表のように0〜3とみなされます。(各カラーコードの下位2ビットが使用するカーラーコードとなります。)

他の画面でのカラーコード	SCREEN 6でのカラーコード
color 0 (&B0000), color 4 (&B0100), color 8 (&B1000), color12(&B1000)	color 0
color 1 (&B0001), color 5 (&B0101), color 9 (&B1001), color13(&B1101)	color 1
color 2 (&B0010), color 6 (&B0110), color10(&B1010), color14(&B1110)	color 2
color 3 (&B0011), color 7 (&B0111), color11(&B1011), color15(&B1111)	color 3

注意! SCREEN 6では、周辺色のX座標の偶数ドットと奇数ドットの色を変えることができます。これはカラーコードの4ビット目を指定することで行なわれ、このためSCREEN 6のカラーコードは、0〜31までが指定できます。(ただし色の種類は4色のままです。)

また、SCREEN 6から他の画面へ変えるとき、色の指定が下位2ビットの繰り返しと見なされているので、画面の色が変わります。ただし、16〜31のときは、そのまま下位4ビットが使われます。

例. SCREEN 6で　　SCREEN 1で
color 3 (&B11) →　color15 (&B1111)
color 4 (&B100) →　color 0 (&B0000)

SCREEN 8の色

SCREEN 8ではカラーコードを使わず、0〜255の色レベルで直接色を指定します。色レベルは緑3ビット、赤3ビット, 青2ビットの計8ビットです。

色レベル=GGGRRRBB　(0〜255)
(GGG:緑　RRR:赤　BB:青)

たとえば、明るい緑は&B11100000(=224)、明るい赤は&B11100(=28)となります。

色レベル=緑レベル(0〜7)*32+赤レベル(0〜7)*4+青レベル(0〜3)

参照 第1章、SCREEN, COLOR SPRITE, 第5章
SCREEN 7と8は、VRAMの容量が128KBの機種のみで使用できます。

COLOR SPRITE

color sprite（カラー・スプライト） MSX2

働　き　SCREEN 4～8の、スプライトの色を指定します。

書き方　COLOR SPRITE（〈スプライト面番号〉）＝〈カラーコード〉

例　COLOR SPRITE(1)

説　明　スプライトの色を指定します。
PUT SPRITE命令で、座標指定とパターン番号の指定を省略した場合と同じ働きですが、〈カラーコード〉にBit 4～5が指定できます。Bit 4～Bit 6については、COLOR SPRITE$を参照してください。

SCREEN 7と8は、VRAMの容量が128KBの機種のみ使用できます。

参　照　COLOR，COLOR SPRITE$，SPRITE$

COLOR SPRITE$

color sprite$（カラー・スプライト・ドル） MSX2

働　き　SCREEN 4～8の、スプライトの色を1ラインごとに指定します。

書き方　COLOR SPRITE$（〈スプライト面番号〉）＝〈式〉

例　COLOR SPRITE$(1)=CHR$(8)+CHR$(15)

説　明　MSX2では、SCREEN 4～8のスプライトの色を1ラインごとに指定することができます。
〈式〉の指定は文字列で行ないます。色指定を指定していない部分は、それ以前にPUT SPRITE命令で指定した色（何も指定していないときは前景色）となります。

例. COLOR SPRITE$(0)=CHR$(2)+CHR$(8)
　　COLOR SPRITE$(0)= "28"

上の2つの例は同じ働きをします。スプライトパターンの1ライン目を緑色にし、2ライン目を赤色にします。3ライン目からは以前に指定した色のままです。
スプライトの色はPUT SPRITE，COLOR SPRITE，COLOR SPRITE$の命令で指定でき、最後に指定した命令が優先されます。
なお、指定する色は0～15のカラーコードで、カラーコードの表示色はCOLOR命令で変えることができます。

SCREEN 7と8は、VRAMの容量が128KBの機種のみが使用できます。

サンプルプログラム

```
10 SCREEN 4,1
20 SPRITE$(1)=STRING$(8,255)
30 SPRITE$(2)=STRING$(8,255)
40 PUT SPRITE 1,(100,100),7
50 PUT SPRITE 2,(150,100),12
60 COLOR SPRITE$(1)=CHR$(8)+CHR$(15)
70 COLOR SPRITE$(2)="28"
80 GOTO 80
```

COLOR SPRITE$

color sprite$ (カラー・スプライト・ドル)　MSX2

Bit 4～Bit 7について

COLOR SPRITE命令のカラーコードが15を超えるとき、上位4ビットは次の意味を持ちます。

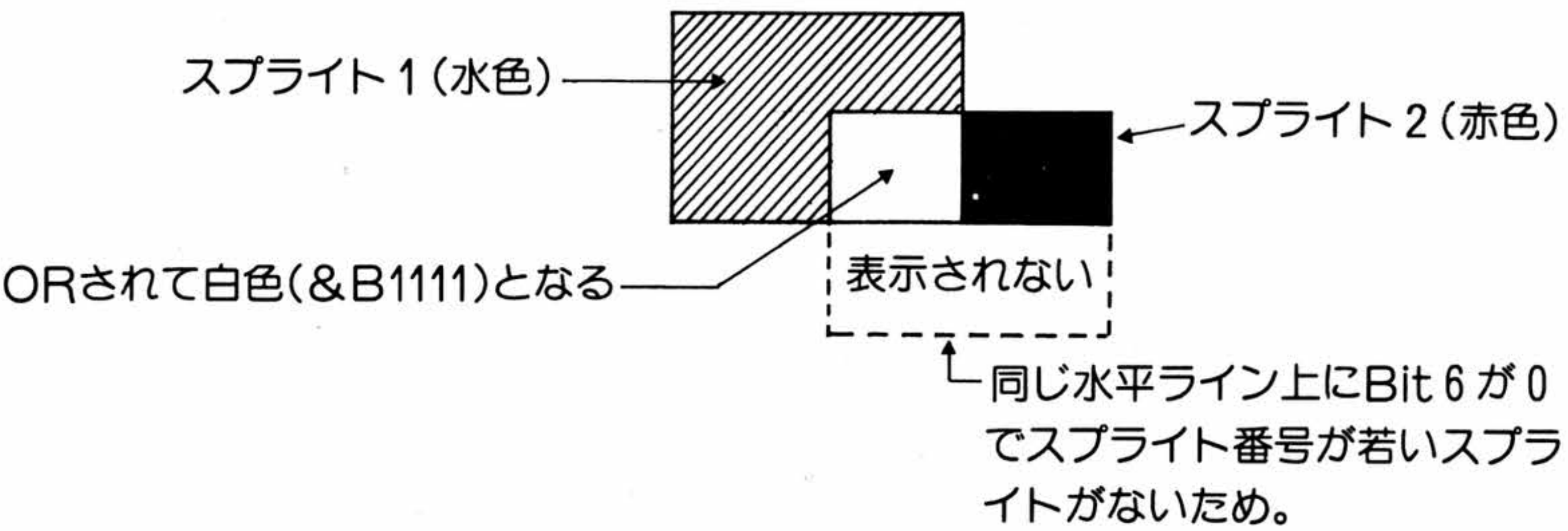
Bit 7　1のときに、そのラインを32ドット左へずらして表示します。

Bit 6　1のパターンが連続しているとき、1つのPUT SPRITE命令でそれ以降に連続しているスプライトを同時に動かします。また、スプライトの重なった部分の色は、カラーコードがORされて表示されます。また、衝突は検出されません。
なお、表示する位置をズラしていると、スプライトの連続移動はできません。

Bit 5　1のとき、衝突を検出しない。

Bit 4　未使用

注意　Bit 6が1のとき、表示が複雑になります。
SRPITE$(1)=STRING$(8,255):SPRITE(2)=STRING$(8,255)を実行し、スプライトの1が&B111(水色)、スプライトの2が&B1001000(赤色)を表示したとき、

スプライト1(水色)
スプライト2(赤色)
ORされて白色(&B1111)となる
表示されない
同じ水平ライン上にBit 6が0でスプライト番号が若いスプライトがないため。

SCREEN 8のスプライト色

SCREEN 8のスプライトだけは、次の16色になっています(変更できません)。

カラーコード	0	1	2	3	4	5	6	7
表示色	黒	暗い青	暗い赤	暗い紫	暗い緑	暗い水色	暗い黄	灰　色
カラーコード	8	9	10	11	12	13	14	15
表示色	肌色	青	赤	紫	緑	水色	黄	白

CONT

continue（コンティニュー：続ける）　コマンド

働き　中断したプログラムの実行を再開する。

書き方　CONT

説明　CTRL ＋ STOP を入力したときや、STOP、END命令を実行して中断したプログラムの実行を再開します。
CTRL ＋ STOP のキー入力やSTOP命令などによりプログラムの実行を中断すると、パソコンはダイレクトモードとなります。この状態でプログラム中の変数の値などを調べたり変更でき、その後CONTを入力すると、中断した次のステートメントから実行を再開します。
INPUT文で中断したときは、そのINPUT文から再開します。
実行を中断したとき、LISTやPRINTなどのコマンドは実行できますが、プログラムの変更（追加、訂正、削除など）を行なうとCONT命令によって実行を再開することはできません。（Can't CONTINE と表示されます。）

C

COPY

Copy（コピー：複写する）　Disk BASIC

働き　ファイルをコピーします。

書き方　COPY 〈ファイルスペック1〉〔TO 〈ファイルスペック2〉〕

例　COPY "B：DEMO" TO "A：MSX"

説明　〈ファイルスペック1〉で指定されたファイルを、〈ファイルスペック2〉で指定されたファイルとしてコピーします。上の例では、ドライブBにある"DEMO"という名前のファイルをドライブAへ"MSX"という名前でコピーします。
コピーできるのはディスクにあるファイルです。カセットや画面のファイルはコピーできません。また、〈ファイルスペック2〉で指定したファイルがすでにあるとき、ファイルは書き変えられます。

ファイルをすべてコピーするときは、次のように指定します。

COPY "B:" TO "A:"

参照　FILES

COPY

COPY（コピー：複写する） MSX2

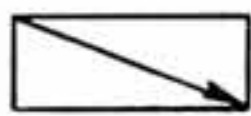
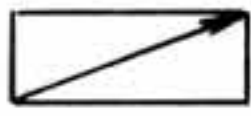
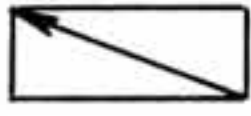
働き VRAM、配列変数、フロッピーディスクのファイルの間で、画像データを転送します。

書き方 ①COPY（〈転送元座標1〉）－（〈転送元座標2〉）〔,〈転送元ページ〉〕TO
（〈転送先座標〉）〔,〈転送先ページ〉〕〔,〈論理演算子〉〕

②COPY（〈転送元座標1〉）－（〈転送元座標2〉）〔,〈転送元ページ〉〕TO
| "〈ファイルスペック〉" |
| 〈配列変数名〉 |

③COPY | "〈ファイルスペック〉" | 〔,〈方向〉〕TO（〈転送先座標〉）
| 〈配列変数名〉 |
〔,〈転送先ページ〉〕〔,〈論理演算子〉〕

④COPY | "〈ファイルスペック〉" | TO | 〈配列変数名〉 |
| 〈配列変数名〉 | | "〈ファイルスペック〉" |

説明 VRAM、宣言した配列変数、フロッピーディスクのファイルの間で、画像データを転送します。
画面モードが、SCREEN 5～8のときに有効です。
書き方①ではVRAM同士で、**書き方②**と**書き方③**ではVRAMと配列変数およびVRAMとフロッピーディスクのファイル間で、そして**書き方④**では配列変数とフロッピーディスクのファイル間で画像データを転送します。
（〈転送元座標1〉）－（〈転送元座標2〉）は、転送元の長方形の領域をLINE命令のように始点、終点として指定します。（〈転送元座標1〉）の位置から転送が始まり、（〈転送元座標2〉）の位置で転送が終わるので、同じ領域を指定するときでも4通りの転送が可能です。

〈方向〉は、画像データを画面に書き込む向きのことで、0から3で指定します。どの方向でも、（〈転送先座標〉）が書き込むデータの開始位置となります。省略値は0です。

方向
- 0 ：左上から右下（省略値）
- 1 ：右上から左下
- 2 ：左下から右上
- 3 ：右下から左上

〈転送元ページ〉は転送元の画面ページを、〈転送先ページ〉は転送先の画面のページを指定します。省略すると、アクティブページとなります（ページについては、SET PAGEを参照してください）。このことから、画面に表れていない画像データのコピーもできます。

〈論理演算子〉は、転送されてきた画像データを、画面に表示されているデータの上にどのように書き込むかを指定するものです。くわしくは、LINE命令を参照してください。

〈配列変数〉に画像データをコピーすることもできます。コピーするときに必要な配列の大きさ（メモリ）は、次のようになります。必要なメモリ（バイト数）は、

N＝((横のドット数）＊　縦のドット数)/K＋5)（端数は切り上げます）

Kは画面のモードによって決まる定数です。また、配列の型によってDIM文で宣言する配列の要素数は次のようになります。

〈Kの値〉		〈配列の型〉	〈要素数〉	
SCREEN 5と7	：2	整数型	：(N＋1)¥2	（端数は切り上げます）
SCREEN 6	：4	単精度型	：(N＋3)¥4	
SCREEN 8	：1	倍精度型	：(N＋7)¥8	

例）

SCREEN 6の画像データを整数型の配列にコピーする場合。

COPY（0，0）－（99，39）TO S%

であれば、横のドット数100（＝99－0＋1）、縦のドット数40（39－0＋1）、K＝4より

N＝(100＊40/4+5)＝1005（バイト）

(1005＋1)¥2＝503（要素として503必要）

となり、配列S%は最低、DIMS%（502）としておかなければエラーとなります。

参　照 SET PAGE，LINE，DIM

サンプルプログラム

```
10 COLOR 15,1,1:SCREEN 5
20 DIM A%(242)
30 CIRCLE(30,30),25,15,-1.57,-3.14
40 PAINT(20,20),12,15
50 COPY(5,5)-(35,35) TO A%
60 CLS:K=0
70 FOR X=0 TO 250 STEP 50
80 FOR Y=0 TO 200 STEP 50
90 COPY A%,(K MOD 3) TO (X,Y)
100 K=K+1:NEXT Y,X
110 GOTO 110
```

ちょっと一言

■論理演算子について、

COPY命令では、LINE命令で使う論理演算子に加えて、転送する画像データ中の透明色の部分を転送しない次の5種類も使えます（色のある部分については、“T”のない論理演算子と同じ働きです)。

TPSET，TPRESET，TXOR，TOR，TAND

COPY SCREEN

copy screen (コピースクリーン：画面をディジタイズする) MSX2

働き テレビやビデオの画面をデータとしてビデオRAMに書き込みます。

書き方 COPY SCREEN〔〈モード〉〕〔,〈マスク〉〕

例 COPY SCREEN 1

説明 この機能は、MSX2のオプション機能です（標準機能ではありません）。
この機能を備えた機種のみに限り有効です。

RGBマルチ端子またはビデオ入力端子から入力されているビデオ信号を赤、緑、青の色レベルに対応したデータとしてビデオRAM(VRAM)に書き込みます(ディジタイズ機能)。
この機能はSCREEN 5～8で使用することができます。

〈モード〉が0の場合、1フィールドのビデオ信号をディジタイズして、ディスプレイページに書き込みます。〈モード〉が省略された場合、0とみなされます。

〈モード〉が1の場合、連続する2フィールド（つまり1フレーム）のビデオ信号をディジタイズして、ディスプレイページの1つ前のページとディスプレイページとに書き込みます。
〈マスク〉はパソコンに書き込む色を指定します。上位3ビットに1が立つと赤が書き込まれ、次の3ビットに1が立つと緑が、下位の2ビットに1が立つと青が書き込まれます。省略値は255です。

SCREEN 7と8はVRAM容量が128KBの機種のみ使用できます。なお、テレビの画面は、1画面（1フレーム）が2つの信号（2つのフィールド）によって構成されています。

参照 COLOR, SET PAGE, SET VIDEO

サンプルプログラム

```
10 '** COPY SCREEN **
20 SET VIDEO 3
30 SCREEN 8:COLOR ,,255
40 COPY SCREEN:T!=TIME
50 IF T!+4>TIME THEN 50
60 SET VIDEO 0
70 IF INKEY$="" GOTO 70 ELSE 20
```

※50行目のプログラムは、画面の安定を得るための待ち時間です。

COS

cosine (コサイン：余弦) 関数

働き 余弦（コサイン）を得る。

書き方 COS (〈数〉)

例 A=COS (45*3.1416/180)　　実行結果。……Aは、0.70710548251122

説明 〈数〉の余弦（コサイン）を得ます。〈数〉の単位はラジアンです。また、得られる結果は倍精度です。

参照 SIN, TAN

CSAVE

c-save（シーセーブ：貯える）　コマンド

働き カセットテープにプログラムを記録します。

書き方 CSAVE "〈ファイル名〉"〔,〈ボーレート〉〕

例 CSAVE "ABC"

説明 CSAVEはパソコンのメモリにあるプログラムに、〈ファイル名〉をつけてカセットテープにセーブ（記録）します。記録はバイナリ形式（短縮されたことば）で行なわれますので、プログラムの合成を行なうときはSAVE命令でセーブします。

〈ファイル名〉は最大6文字からなる文字列で指定します。
〈ボーレート〉は、カセットテープへ記録するときの速度です。1、2の2種類が指定でき、省略するとSCREEN文で指定した〈ボーレート〉でセーブされます。
　1：1200ボー
　2：2400ボー
CSAVEを行なった後、CLOAD?文でプログラムが正確にセーブできたかを確認してください。
機械語のプログラムはBSAVE，BLOADを使ってセーブ、ロードします。

参照 BSAVE, CLOAD, SAVE, SCREEN

CSNG

c-single（シーシングル：単数の）　関数

働き 数値を単精度に変換します。

書き方 CSNG（〈数〉）

例 A！=CSNG（B＃）

説明 〈数〉の値を6桁の単精度実数型に変換します。（7桁目を四捨五入しています。）
精度が高くなくてもよい計算やメモリを節約したいときに使います。
CSNGでの変換は、単精度実数型変数への代入と同じ結果になります。

参照 CINT，CDBL

CSRLIN

cursor line（カーソルライン：カーソルの行）　　システム変数

働き カーソルの垂直位置を得る。

書き方 CSRLIN

例 Y＝CSRLIN

説明 現在のカーソルの垂直位置を0～22（KEY OFFで0～23）の行単位で得ます。なお、画面の最上行が0で値が大きくなるほど画面の下部になります。
システム変数なので値を代入することはできません。また、テキスト画面のみで使用できます。
カーソルの水平位置（桁）を知るためにはPOS関数を使います。

参照 POS, LOCATE

CVI／CVS／CVD

CVI/CVS/CVD（インテジャー/シングル/ダブル）　　Disk BASIC

働き 文字列を数値データに変換します。

書き方 ① CVI（《2バイト文字列》）
② CVS（《4バイト文字列》）
③ CVD（《8バイト文字列》）

例 ① A%＝CVI（"AF"）
② B！＝CVS（"D800"）
③ C＝CVD（AB$）

説明 ディスケットのランダムアクセスファイルから読み込んだ数値データは、文字として記録されているため、これらの関数を使って数値のデータに変換しなければなりません。
CVIは2文字（2バイト＝16ビット）の文字列を整数値に、CVSは4文字の文字列を単精度実数値に、CVDは8文字の文字列を倍精度実数にそれぞれ変換します。

参照 MKI$／MKS$／MKD$

サンプルプログラム

```
10 OPEN "MSX" AS #1
20 FIELD #1,2 AS A$,4 AS B$,8 AS C$
30 A%=5:B!=3.14:C#=1.2345678901#
40 LSET A$=MKI$(A%)
50 LSET B$=MKS$(B!)
60 LSET C$=MKD$(C#)
70 PUT #1,1
80 CLOSE #1
90 '
100 OPEN "MSX" AS #1
110 FIELD #1,2 AS D$,4 AS E$,8 AS F$
120 GET #1,1
130 PRINT CVI(D$)
140 PRINT CVS(E$)
150 PRINT CVD(F$)
160 CLOSE #1
170 END
```

（行20～60）← 整数、単精度実数、倍精度実数を文字として記録します。

（行110～150）← 文字として記録されている整数、単精度実数、倍精度実数を数値に変換します。

DATA

data（データ：情報） ステートメント

働き READ文で読み出すデータをプログラム中に用意する。

書き方 DATA 〈データ〉〔,〈データ〉…〕

例 DATA 12, AB, "34.56"

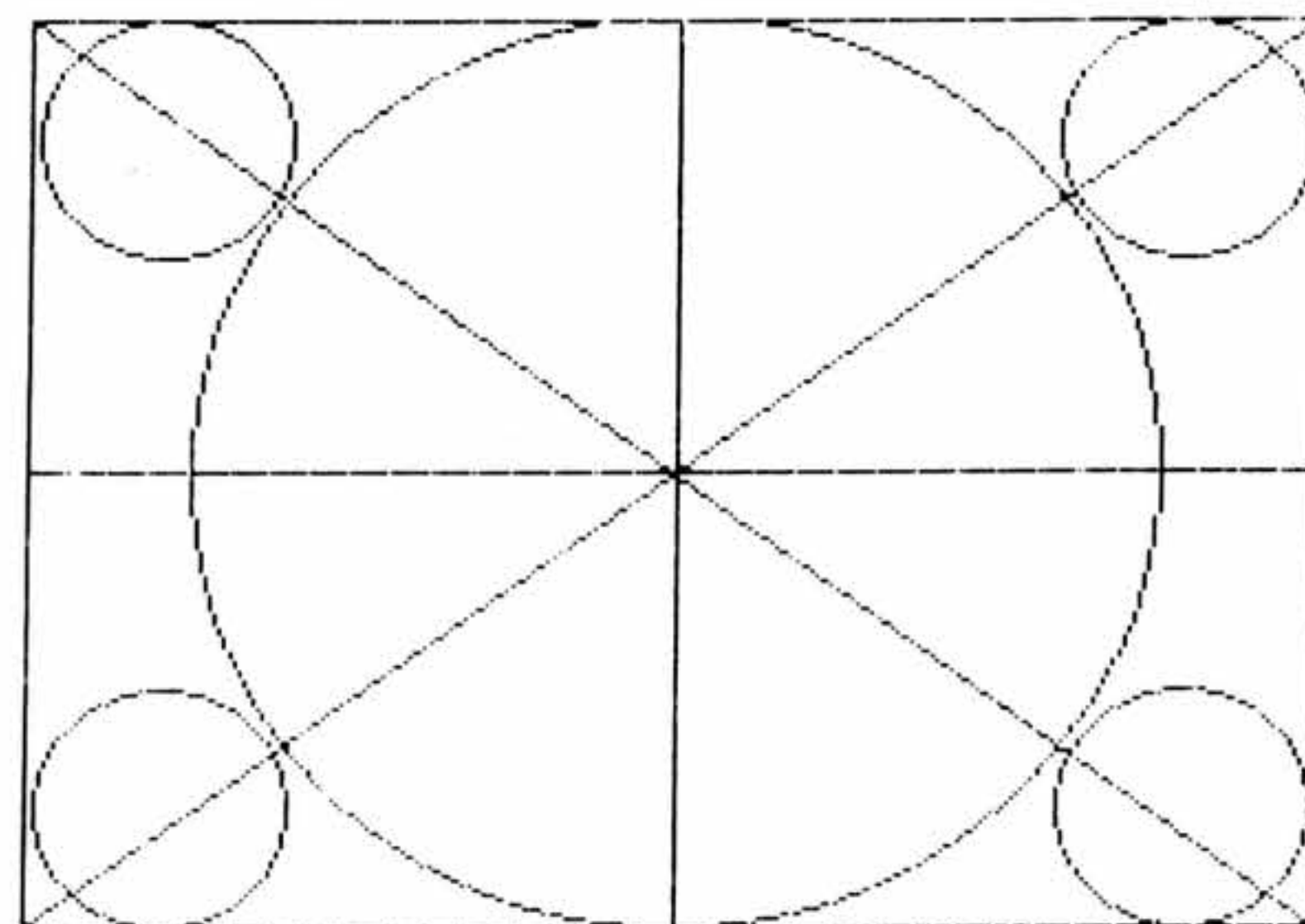
説明 DATA文はREAD文で読み出す〈データ〉を用意します。
〈データ〉は、プログラム中の計算や表示に必要な数や文字列です。〈データ〉は1行（255文字）に入る範囲で、カンマで区切っていくつでも指定できます。
1つのプログラム中には複数のDATA文を指定できますが、〈データ〉はプログラム内で連続したデータの並びとみなされます。
RESTORE文を使うとREAD文で読み出すDATA文を行単位で指定できます。
DATA文はプログラムで実行されない（非実行文）ので、プログラム中どこに設定してもかまいません。
DATA文の中の文字列に「，」(カンマ）や文字列の先頭または最後にスペースや「・」(ピリオド）を含むときは、それぞれの文字列を「"」(ダブルクオーテーション）で囲まなければなりません。

〈データ〉が数値のとき、整数、単精度実数、倍精度実数、文字列のいずれでもかまいませんが、式（9＊3など）は許されません。また、READ文の変数の型とDATA文の〈データ〉の型が一致しなければなりません。

参照 READ, RESTORE

サンプルプログラム

ラインとサークルを描きます。

```
10 '** LINE+CIRCLE**
20 COLOR 15,1,1:SCREEN 2
30 LINE(0,0)-(255,191),15,B
40 READ A,B,C,D:IF A<0 THEN 60
50 LINE(A,B)-(C,D),15:GOTO 40
60 READ A,B,C:IF C=0 THEN 120
70 CIRCLE(A,B),C,15:GOTO 60
80 DATA 0,0,255,191,0,191,255,0,0,96
90 DATA 255,96,127,0,127,191,-1,0,0,0
100 DATA 127,96,100,127,96,50,25,25,25
110 DATA 25,166,24,230,25,25,230,166,25,
0,0,0
120 GOTO 120
```

DEF FN

define function（デファイン・ファンクション：関数を定義する）ステートメント

働き ユーザー関数の定義をします。

書き方 DEF FN 〈名前〉〔(〈引数〉〔,〈引数〉…〕)〕=〈関数の定義式〉

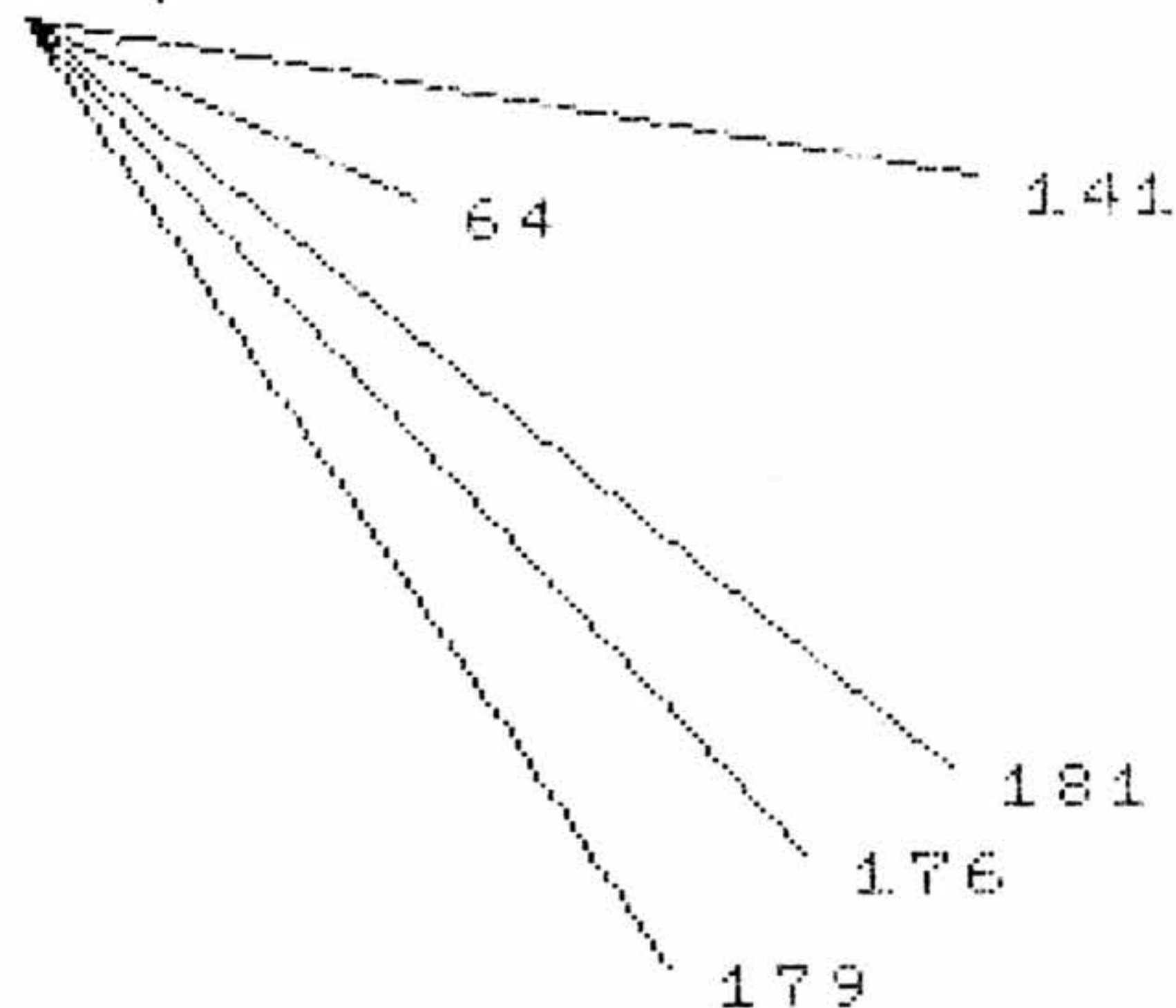
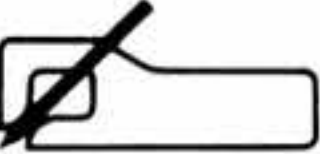
例 DEF FNA(X, Y) = SQR(X＊X+Y＊Y)

説明 プログラム中でたびたび使う計算式をあらかじめ関数として定義しておくと、以降のプログラムではいちいち計算式を設定しなくても関数を呼び出すだけで計算ができます。
(プログラムを組む人が自由に定義できるので、これをユーザー関数という)

FNとこれに続く1～2文字の英数字により定義する関数の名前を設定します。
〈引数〉は〈関係の定義式〉の中で使っている同じ名前の変数に対応します。〈引数〉は関数の定義のときにのみ使われるだけなので、以降で同じ名前の変数を使っても影響はありません。
〈関数の定義式〉は関数の計算内容を定義する式で、1行で設定し、関数を呼び出すときに指定する〈引数〉の型と〈関数の定義式〉の中の〈引数〉とは型が同じでなければなりません。
DEF FN文は関数が呼び出される前に定義します。

サンプルプログラム

```
10 '** ｷｮﾘ ｦ ﾓﾄﾒﾙ **
20 DEF FNA(X,Y)=CINT(SQR((X-10)^2+(Y-10)
^2))
30 COLOR 15,1,1:SCREEN 2
40 OPEN "GRP:" FOR OUTPUT AS #1
50 DRAW"BM5,0":PRINT #1,"(10,10)ｶﾗﾉ ｷｮﾘ"
60 FOR S=1 TO 5:X=RND(1)*200+30
70 Y=RND(1)*150+20:COLOR RND(1)*13+2
80 LINE(10,10)-(X,Y),15
90 DRAW"BM=X;,=Y;":PRINT #1,FNA(X,Y)
100 NEXT S:CLOSE
110 GOTO 110
```

(10,10)ｶﾗﾉ ｷｮﾘ
64
141
181
176
179

DEFINT／SNG／DBL／STR

define integer（デファイン インテジャー）
define single　（デファイン シングル）
define duble　 （デファイン ダブル）
defin string　（デファイン ストリングス）

ステートメント

働　き 変数の型を宣言します。

書き方
DEFINT　〈文字の範囲〉〔,〈文字の範囲〉…〕
DEFSNG　〈文字の範囲〉〔,〈文字の範囲〉…〕
DEFDBL　〈文字の範囲〉〔,〈文字の範囲〉…〕
DEFSTR　〈文字の範囲〉〔,〈文字の範囲〉…〕

例　DEFINT　A−D, X, Y

説　明 指定した頭文字の範囲で始まる変数、配列変数の型を次のように宣言します。
〈文字の範囲〉は〈英字1文字〉または〈英字1文字—英字1文字〉で指定します。

DEFINT …. 整数型　　例：DEFINT　A−D, X, Y
　　　　AからD、およびX，Yで始まる変数が整数型
DEFSNG …. 単精度型　例：DEFSNG　K, I
　　　　K，Iで始まる変数が単精度型
DEFDBL …. 倍精度型　例：DEFDBL　X, Y
　　　　X，Yで始まる変数が倍精度型
DBLSTR …. 文字型　　例：DEFSTR　A−G
　　　　AからGで始まる変数が文字型

型の宣言を行なわなかった文字で始まり属性文字（%、！、#、$など）を持たない変数はすべて倍精度型となります。
また、変数に属性文字をつけた場合、変数の型はDEF命令での型宣言よりも属性文字の指定が優先します。

注意！ DEFと〈型〉（INTなど）の間は、スペースをあけるとSyntax errorとなります。また、「−」でつないだ文字はアルファベット順でないとエラーとなります。
悪い例　• DEF　INT　A−B　←DEFとINTの間のスペース
　　　　• DEFSNG　Z−R　←アルファベット順でない

DEF文で宣言した後にCLEAR文を実行するとDEF文での宣言はすべて無効となります。

DEF USR

define user（デファインユーザ:使用者が設定する）　ステートメント

働　き 機械語で作られたプログラムの実行開始番地を設定します。

書き方 DEF　USR〔〈プログラム番号〉〕=〈開始番地〉

例 DEF　USR=&H8010

説　明 USR関数（ユーザー関数）が呼び出す機械語プログラムの実行開始番地を設定します。
〈プログラム番号〉は0～9の値で、複数のプログラムを用いることができます。省略した場合は0とみなされます。
DEF　USR文は2重に設定してもエラーになりません。後の設定が有効となります。

ちょっと一言 DEF　USR文は、USR関数と対応して使われます。DEF　USR文で機械語プログラムの開始番地を設定し、USR関数で実行します。
機械語プログラムは、BASICプログラムのサブルーチンとして使われる場合と、BASICで実行開始した後は全て機械語で実行する場合とがあります。

参　照 USR，CLEAR

DELETE

delete（デリート:削除する）　コマンド

働　き プログラムの一部を消します。

書き方 DELETE〈始点行番号〉〔-〈終点行番号〉〕
DELETE　-〈終点行番号〉

例
DELETE　100-200　実行結果……100行から200行のプログラムを消します。
DELETE　-200　実行結果……最初の行から200行までのプログラムを消します。

説　明 〈始点行番号〉から〈終点行番号〉までのプログラムを消します。
〈始点行番号〉だけを指定したときは、その行だけを消します。
-〈終点行番号〉を指定すると、プログラムの先頭から〈終点行番号〉までを消します。

DELETEの後に“.”（ピリオド）を付けると、現在実行を停止している行が消されます。また、〈始点行番号〉が存在しないときは、その次の行から消されますが、〈終点行番号〉が存在しないとIllegal function callエラーとなります。

DIM

dimension（ディメンジョン:次元・大きさ）　　ステートメント

働　き **配列変数を定義し、メモリを割り当てます。**

書き方 DIM **〈変数名〉(〈添字の最大値〉〔,〈添字の最大値〉…〕)**

例 DIM　A(15,15), B(7)

説　明 配列変数の次元の数と添字の最大値を設定します。
配列の次元数はそれぞれの〈添字の最大値〉の数によって決定され、〈添字の最大値〉の数だけメモリに配列の領域を割り当てます。

添字の最小値は0です。また、DIM文で宣言しないで配列変数を用いたとき、〈添字の最大値〉は10に設定されます。

DIM文が実行された時点でその配列のすべての要素（データ）は0（文字型のときはヌルストリング）に設定されます。
不用になった配列変数はERASE文またはCLEAR文で削除できます。
また、一度宣言してある配列変数は、ERASEまたはCLEARで削除しないと再宣言できません。

参　照 第1章、CLEAR, ERASE

DRAW

draw（ドロー：描く）　　ステートメント

働　き グラフィック画面に図形を描く

書き方 DRAW〈文字式〉

例　DRAW "R100D100H100"

説　明 DRAWとこれに続く作画命令により画面に図形（線）を描きます。DRAW文はグラフィックモードで使用します。

DRAW文には次のような作画命令（グラフィックマクロ命令）があります。

● **移動マクロ命令**

U〈距離〉	上へ移動
D〈距離〉	下へ移動
L〈距離〉	左へ移動
R〈距離〉	右へ移動
E〈距離〉	右斜め上へ移動
F〈距離〉	右斜め下へ移動
G〈距離〉	左斜め下へ移動
H〈距離〉	左斜め上へ移動

H　U　E
L　○　R
G　D　F

移動命令は、基準点（参照点）からそれぞれの方向に指定された距離だけ作画します。

Mx，y　移動位置をX方向、Y方向についての絶対座標あるいは相対座標で指定します。ここで、Xの前にプラス(＋)あるいはマイナス(－)が書かれていれば、参照点からの相対座標となります。

例）　DRAW "M100，100M+50，+50"

参照点(LP)から(100,100)へ線を描き、(100,100)から(150,150)へ線を描く

B　各々の方向へ移動しますが、作画は行いません。U～Mの命令の直前に付けます。

例）　DRAW "BM100,100M+50,+50"

(100，100)から(150，150)へ線を描きます

N　各々の方向へ作画しながら移動しますが、作画後の参照点は支点となります。U･Mの命令の直前に付けます。

例）　DRAW "M100,100NM+50,+50M+50,-50"

参照点(LP)から(100,100)へ線を描き、(100,100)から(150,150)へ線を描きます。

その後、(LP)から(右へ50,下へ50の位置)へ線を描きます。

移動マクロ命令の〈距離〉は、各方向への移動距離をドット単位で指定しますが、実際の移動距離は、〈距離〉にS命令で指定されるスケールをかけ合せたものです。また、〈距離〉は同じでも方向によって移動距離が異なります。

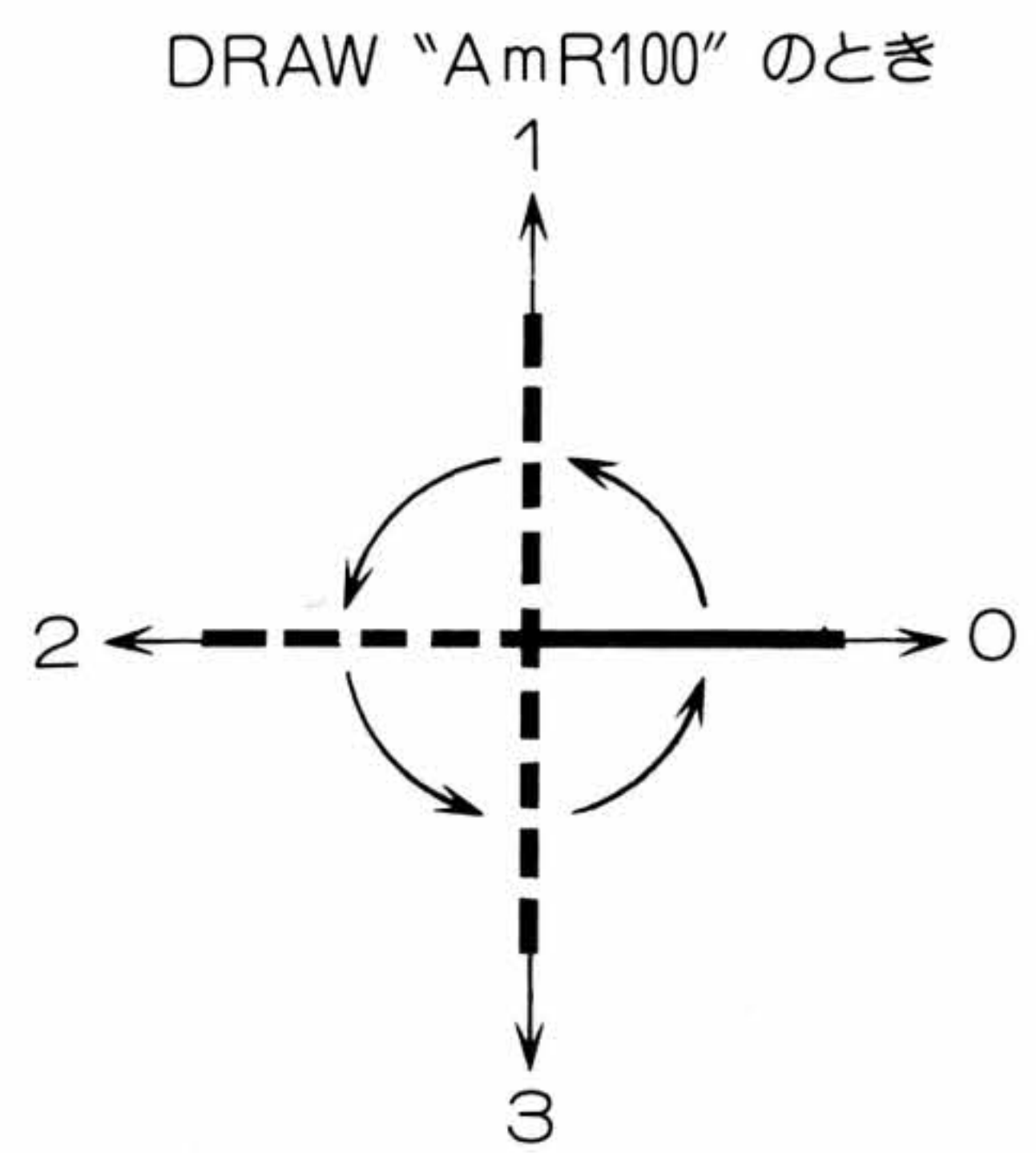
● **回転マクロ命令**

A〈角度〉　U～Mの移動マクロ命令で描く図形を90°単位で回転して表示します。（M命令は相対座標を指定するときのみ）

〈角度〉は0～3の整数で表し、0は0度、1は90度、2は180度、3は270度を表します。

1度指定すると、次のA命令まで指定した回転の状態が続きます。

DRAW "AmR100" のとき

1
2
0
3

● **色指定マクロ命令**

C〈カラーコード〉　〈カラーコード〉指定した色で図形を描きます。〈カラーコード〉は0～15の整数です。

● **スケールマクロ命令**

S〈倍率〉　図形の大きさを決めます。〈倍率〉0～255の整数で、この値の1/4が実際の倍率となります。たとえば〈倍率〉に1を指定したとき、実際は1/4倍となります。

U～Mの〈距離〉に倍率をかけ合せたものが移動距離となります。(M命令は相対座標を指定するとき)。

S命令を指定すると、次のS命令まで同じ倍率となります。また、〈倍率〉の省略値は4(倍率1)です。

● **ローカルマクロ命令**

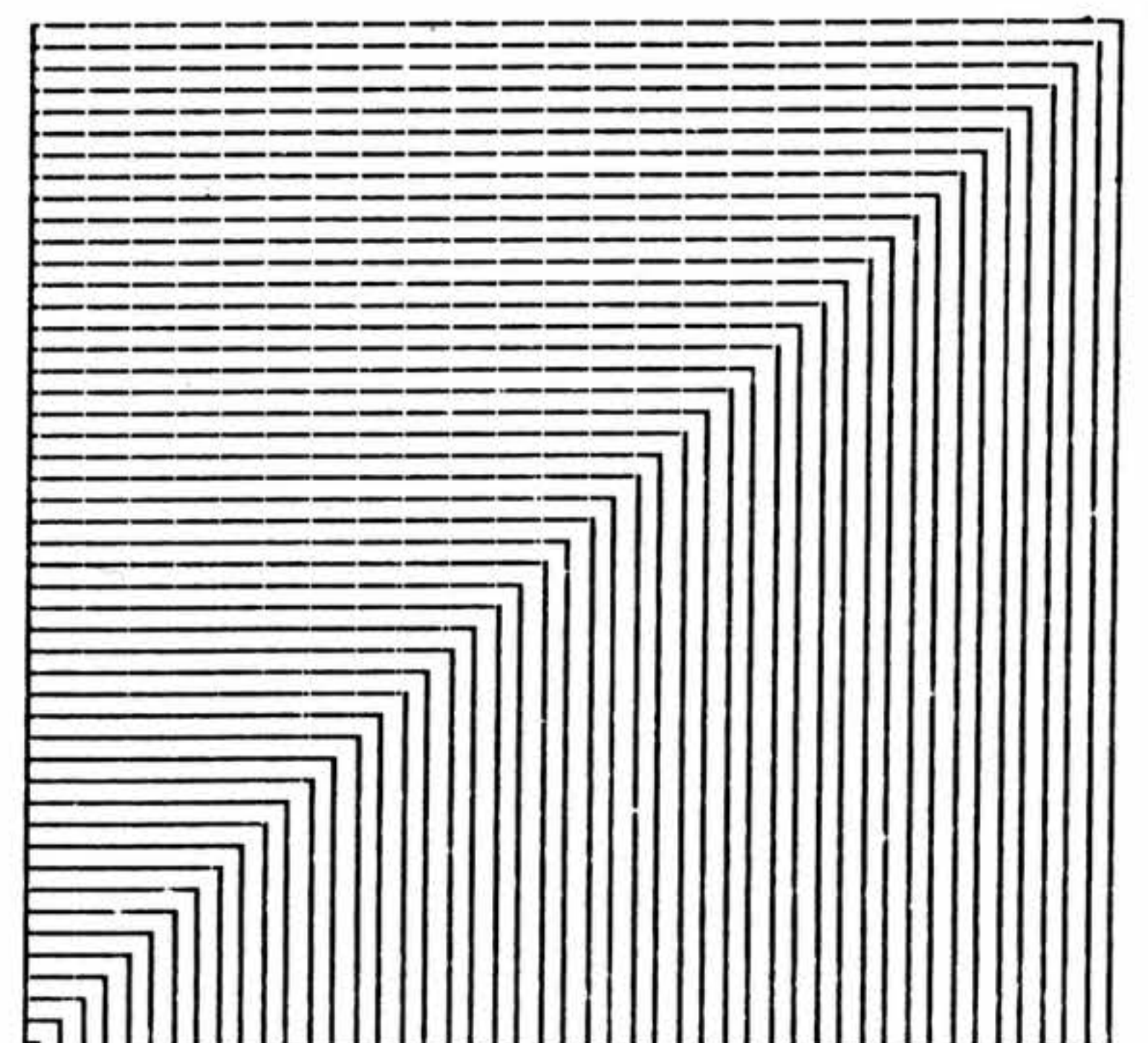
X〈文字変数〉;　〈文字変数〉の文字列に含まれている作画命令を実行します。〈文字変数〉の後のセミコロン(;)必ず記入してください。
X命令は、作画命令が255文字を越えるときや、繰り返し図形を描くときなどに便利です。す。

例)　B$="E50F50"; DRAW "XB$;XB$;XB$;"

以上の作画命令の〈距離〉や〈カラーコード〉などには、定数のほかに "=〈変数名〉;" とすることで、数値変数を指定できます。(";" は必ず記入します。)

例)　A=50：B=100：DRAW "M=A;,=B;"

サンプルプログラム

```
10 COLOR ,1,1:SCREEN 2:C=15
20 PSET(30,190)
30 FOR K=190 TO 1 STEP -4
40 COLOR C
50 A$="R"+STR$(K)+"U"+STR$(K)+"L"+STR$(K
)+"D"+STR$(K)
60 DRAW"XA$;"
70 C=C-1:IF C=1 THEN C=15
80 NEXT K
```

DSKF

disk File（ディスク ファイル）　　Disk BASIC

働 き　フロッピーディスクの残り容量をクラスタ単位で得ます。

書き方　DSKF（〈ドライブ番号〉）

例　PRINT　DSKF（1）

説 明　〈ドライブ番号〉で指定されたフロッピーディスクの書き込み可能な残り容量をクラスタ単位で得ます。

ドライブ番号とドライブ名は次のように対応します。
0ーデフォルトドライブ
1ードライブA
2ードライブB

3以降は，3…C，4…Dと順に対応します。

用 語　1クラスタは、3.5インチフロッピーディスクの場合2セクタです。1セクタに512バイトのデータが記録できます。

END

end（エンド:終了）　　ステートメント

働 き　プログラムの実行を終了します。

書き方　END

説 明　プログラムの実行を終了し、すべてのファイルを閉じた後にコマンドの入力持ち（コマンドレベル）に戻ります。

END文はプログラム中にいくつ書いてもかまいません。また、プログラムの最後のEND文は省略することができますが、このときファイルは閉じられません。

サンプルプログラム

```
10 PRINT "ﾌﾟﾛｸﾞﾗﾑ ﾉ ｼｭｳﾘｮｳ "
20 END
30 PRINT " ｼﾞｯｺｳ ｻﾚﾏｾﾝ"
```

参 照　STOP，CLOSE

EOF

end of file（エンド オブ ファイル：ファイルの終了）　関数

働き　ファイルが終了したかどうかを調べます。

書き方　EOF（〈ファイル番号〉）

例　IF EOF(1) THEN CLOSE #1　……ファイルが終了するとCLOSEする。

説明　〈ファイル番号〉で指定したファイルのデータが終了しているかを調べます。終了しているときは－1を、そうでなければ0を得ます。

〈ファイル番号〉で指定されたファイルはOPEN命令ですでに開かれていなければなりません。

注意！ ファイル中のデータを読み出し、終了した後にINPUT命令などでデータを読み出そうとすると、"Input past end" エラーとなります。

参照　OPEN，INPUT$

```
10 OPEN "DEMO" FOR OUTPUT AS #1
20 PRINT #1,"A B C D E F "
30 CLOSE:END
40 OPEN "DEMO" FOR INPUT AS #1
50 INPUT #1,A$:PRINT A$
60 IF EOF(1)<>-1 THEN 50
70 CLOSE:END
```

D

E

ERASE

erase（イレース：消す）　ステートメント

働き　プログラム中の配列を消去します。

書き方　ERASE 〈配列名〉〔,〈配列名〉…〕

例　ERASE X，Y

説明　DIM文で指定した配列をメモリから消去します。

この命令により、配列に割り当てられていたメモリを別の目的に使うことができます。（メモリ不足のときに使用することもできます。）

また、1度ERASE文により消去された配列をDIM文により再び宣言することができます。ERASEしない配列を2度宣言するとRedimensioned arrayエラーとなります。

参照　DIM，CLEAR

用語　Redimensioned array（レディメンションド アレイ）
「再び配列を宣言した」という意味です。

ERL

error Line（エラーライン:エラー発生行） システム変数

働き エラーの発生した行番号を得ます。

書き方 ERL

例 L＝ERL

説明 エラーが発生したときの行番号を得ます。コマンドの入力持ち（コマンドレベル）でエラーが発生したときは65535を得ます。

一般にERLはERRとともに、エラー処理ルーチンで使用します。

注意！ IF文でERL関数の値を調べるとき、行番号を演算子（＝，＜，＞）の右側に書いてください。左側に書くと、RENUM文を実行したとき行番号が修正されません。

用語 エラー処理ルーチン
エラーが発生したときプログラムの実行を移し、エラーの種類、エラーの発生した行番号などをもとにエラーの対応を行ない、プログラムの実行が中断しないように組まれたプログラムです。

ERR

error（エラーの種類） システム変数

働き 発生したエラーのエラーコードを得ます。

書き方 ERR

例 E＝ERR

説明 エラーが発生したときのエラーコードを得ます。エラーコードとその意味は「エラーメッセージ表」をご覧ください。

参照 ON ERROR GOTO，資料「エラーメッセージ表」

サンプルプログラム

10のかけ算を表示します。

```
10 ON ERROR GOTO 60
20 A=1
30 A=A*10
40 PRINT A
50 GOTO 30
60 IF ERR <>6 THEN 100
70 PRINT "ERL=";ERL
80 PRINT "ERROR ｼｮﾘ ｼﾏｼﾀ"
90 RESUME 20
100 ON ERROR GOTO 0
```

ERROR

error（エラー:エラー状態） ステートメント

働　き　プログラム中で、エラーを発生させます。

書き方　ERROR〈エラーコード〉

例　ERROR　100

説　明　〈エラーコード〉で指定したエラーが発生した状態を作りだします。〈エラーコード〉を指定すると、対応するエラーメッセージがプリントされ、コマンド入力持ち（コマンドレベル）となります。

〈エラーコード〉は０～255の整数で指定します。23および26～45、60～255は未定義のエラーのためUnprintable errorとプリントされます。

ちょっと一言　ERROR文は、ON　ERROR　GOTO文、ERL，ERR文などとともに使い、エラー処理を行います。
ERROR文はBASICではエラーとはならないが、ユーザーが独自のエラーを設定したいときに使います。

例）　試験の平均点が100点以上になったとき、エラーとして、その原因を探す。

参　照　ON　ERROR　GOTO，ERL，ERR，資料「エラーメッセージ表」

EXP

exponential（エクスポネンシャル:指数の） 関数

働　き　eのべき乗を得る

書き方　EXP（〈数〉）

例　PRINT EXP(1)　　実行結果……2.7182818284588

説　明　eのべき乗（〈数〉の回数だけeをかけ算）した結果を得ます。eは自然対数の底です。

$EXP(X)=e^{x}$……e^X　　e：2.718281828…

〈数〉の値は145.06286085862以下でなければなりません。これは、計算結果が9.99999E＋62以上となると、Overflowエラーとなるためです。

〈数〉の型にかかわらず、得られる値は倍精度型実数です。また関数なので他の命令と組合せて使います。

サンプルプログラム

0.5から3.5までのeのべき乗を求める。

```
10 PRINT "   X   EXP(X) LOG(EXP(X))"
20 FOR X=.5 TO 3.5 STEP .3
30 PRINT USING"##.#   ##.#   ##.# ";X,EXP(
X);LOG(EXP(X))
50 NEXT
```

FIELD

field（フィールド：範囲） Disk BASIC

働　き ランダムアクセスファイルの入出力用バッファに、変数領域を割り当てます。

書き方 FIELD〔#〕〈ファイル番号〉,〈フィールド幅〉ＡＳ〈文字変数〉
〔,〈フィールド幅〉ＡＳ〈文字変数〉……〕

例　FIELD #1, 10 AS A$, 20 AS B$

説　明 OPEN文で開いた〈ファイル番号〉のファイルに、文字変数の領域を割り当てます。

FIELD文は、PUT文、GET文を使ってランダムアクセスファイルの入出力を行なう前に実行します。

FIELD文では、ランダムアクセスファイルの入出力用バッファにデータを設定しません。データのセットはLSET、RSET文で行なわれます。

例　FIELD #1, 10 AS A$, 20 AS B$

ランダム入出力用バッファの最初の10文字(バイト)をA$という変数に割り当て、つぎの20文字をB$に割り当てます。

サンプルプログラム

```
10 '** TANGO FILE **
20 OPEN"ETANGO" AS#1
30 FIELD #1,20 AS A$,20 AS B$
40 RC=LOF(1)/255
50 INPUT"ｴｲﾀﾝｺﾞ";T$
60 IF T$="END" OR T$="end" THEN 110
70 INPUT"ﾆﾎﾝｺﾞ";N$
80 LSET A$=T$:LSET B$=N$
90 RC=RC+1:PUT #1,RC
100 GOTO 50
110 PRINT "ｴｲﾀﾝｺﾞ => ﾆﾎﾝｺﾞ"
120 LN=LOF(1)/255
130 INPUT"ｴｲﾀﾝｺﾞ";T$
140 T$=LEFT$(T$+STRING$(20," "),20)
150 FOR RC=1 TO LN
160 GET #1,RC
170 IF T$=A$ THEN PRINT"ﾆﾎﾝｺﾞ ";B$:GOTO
200
180 NEXT RC
190 PRINT "ﾆﾎﾝｺﾞ ｱﾘﾏｾﾝ"
200 CLOSE #1:END
```

参　照 OPEN, LSET/RSET, CVI/CVS/CVD

FILES

files（ファイルズ：ファイル） Disk BASIC

働　き　フロッピーディスク中のファイル名を表示します。

書き方　FILES〔"〈ファイルスペック〉"〕

例　FILES

説　明　〈ファイルスペック〉で指定されたファイルのファイル名を表示します。指定したファイルがフロッピーディスク中になければ "File not found" エラーとなります。

〈ファイルスペック〉を省略すると、ドライブA中の全ファイル名を表示します。
〈ファイルスペック〉でディスクドライブが指定されていれば、そのディスクドライブが選択され、そうでないとき（ファイル名だけを指定したとき）は、ドライブAが選択されます。

例）　FILES "A：" ……ドライブAの全ファイルを表示。

指定するファイル名には疑問符（?）を含むことができ、ファイル名の1文字の代わりとして使えます。

例）　FILES "DE??" ……DEを含む4文字のファイルを全て表示。

指定するファイル名の代わりとしてアスタリクス（*）を含むことができ、不明確なファイル名の検出に利用できます。

例）　FILES "DE*" ……DEで始まるファイルを全て表示。
　　　FILES "*.BAS" ……拡張子BASのついた全ファイルを表示。

FIX

fix（フィックス：整える） 関数

働　き　数値の整数部を得ます。

書き方　FIX（〈数値〉）

例　PRINT FIX（10/3）　　実行結果……3

説　明　〈数値〉の小数点以下を取り去った整数部分を得ます。

FIX（X）はSGN（X）* INT（ABS（X））と同じです。

ちょっと一言

FIX関数とINT関数は〈数値〉が正のとき同じ結果となりますが、負のとき次のように異なります。
FIX関数……〈数値〉の整数部分にマイナス(−)をつけたもの。
INT関数……〈数値〉よりも絶対値が大きくなる。

サンプルプログラム

```
10 PRINT "  A  INT(A) FIX(A)"
20 FOR A=-1.5 TO 1.5 STEP .3
30 B=INT(A):C=FIX(A)
40 PRINT USING"##.#  ##.#  ##.# ";A;B;C
50 NEXT
```

```
   A  INT(A) FIX(A)
 -1.5  -2.0  -1.0
 -1.2  -2.0  -1.0
 -0.9  -1.0   0.0
 -0.6  -1.0   0.0
 -0.3  -1.0   0.0
  0.0   0.0   0.0
  0.3   0.0   0.0
  0.6   0.0   0.0
  0.9   0.0   0.0
  1.2   1.0   1.0
  1.5   1.0   1.0
```

F

FOR～NEXT

for～next（フォー～ネクスト：～から，次へ）　　ステートメント

働　き FORからNEXTまでの間にあるプログラムを指定した回数だけくり返し実行します。

書き方 FOR〈変数名〉=〈初期値〉　TO〈終値〉〔STEP〈増分〉〕
　　∫
NEXT〔〈変数名〉〔,〈変数名〉〕〕…

例

```
FOR N=0 TO 255
 ∫
NEXT N
```

説　明 〈変数名〉の値は、FOR文からNEXT文までのくり返し実行回数のカウンタとして使われ、初めの値は〈初期値〉に設定されます。そして、FOR文からNEXT文までのプログラムを実行し、カウンタの値はSTEP文によって指定された〈増分〉だけ加算（減算）されます。次にカウンタの値を〈終値〉と比べて、〈終値〉に達していなければFOR文の次の文へ戻り、同じプログラムを実行します。

〈初期値〉、〈終値〉、〈増分〉は数値または数式で指定します。
〈増分〉は正の数でも負でもかまいませんがSTEP文が省略されたとき、〈増分〉は＋1とみなされます。

FOR～NEXT文は入れ子（FOR～NEXT文の中にFOR～NEXT文を含む）にできます。このとき、それぞれの〈変数名〉は別のものを使い、1つのFOR～NEXT文はほかのFOR～NEXT文の中に入っていなければなりません。

よい例

```
10 FOR M=0 TO…… ─┐
20 FOR N=0 TO……┐ │
                 │ │
80 NEXT N ───────┘ │
90 NEXT M ─────────┘
```

悪い例

```
10 FOR M=0 TO…… ─┐
20 FOR N=0 TO……┐ │
                 │ │
80 NEXT M ───────┼─┘
90 NEXT N ───────┘
```

入れ子にしたFOR～NEXT文が同じところでNEXT文となるとき、1つのNEXT文にまとめることができます。そのときは、NEXT文に続けて内側のFOR～NEXT文の〈変数名〉から順にカンマ（，）で区切って書きます。

また、NEXT文の〈変数名〉を省略することができます。しかし、〈変数名〉は最も近くのFOR文に対応しているとみなされますので、FOR～NEXT文が入れ子のときには〈変数名〉を指定した方が間違いは少ないでしょう。

例

```
10 FOR M=0 TO…… ─┐
20 FOR N=0 TO……┐ │
                 │ │
80 NEXT N, M ────┴─┘
        ↑
        └順番を間違えないで！
```

例

```
10 FOR M=0 TO…… ─┐
20 FOR N=0 TO……┐ │
                 │ │
80 NEXT ─────────┘ │
90 NEXT ───────────┘
```

注意！ 〈初期値〉、〈終値〉、〈増分〉の数により、FOR～NEXT間のプログラムの実行回数が変わります。

〈初期値〉と〈終値〉	〈増分〉	実行
〈初期値〉が〈終値〉より大きい	正の数	1回だけ実行する
〈初期値〉が〈終値〉より小さい	負の数	1回だけ実行する
〈初期値〉と〈終値〉が等しい	0	何回も繰返す
〈初期値〉と〈終値〉が等しい	0でない数	1回だけ実行する

サンプルプログラム

```
10  FOR A=1 TO 10
20 PRINT A;
30 FOR S=1 TO A
40 PRINT "*";
50 NEXT S:PRINT
60 NEXT A
```

```
run
 1 *
 2 **
 3 ***
 4 ****
 5 *****
 6 ******
 7 *******
 8 ********
 9 *********
 10 **********
```

_FORMAT

format（フォーマット：型、構成）　Disk BASIC

働き ディスケットを初期化する

書き方 CALL FORMAT

説明 Disk BASICが起動されているときに、Disk BASICで使用できるようにディスケットを初期化します。

CALL命令の代りにアンダーバー(_)を使用できます。
FORMATの処理は対話的に進められます。

例

```
_FORMAT [リターン]  ←FORMAT文の入力
Drive name?(A, B) [A]  ←ドライブの指定(例A)
Strike a key when ready  ←何かキーを押すと開始
Format complete  ←フォーマット終了
OK
```

FRE

free（フリー：自由な、規則にとらわれない）　　関数

働　き メモリの未使用領域の大きさを得ます。

書き方 FRE（〈数値〉）
FRE（〈文字列〉）

例　FRE（0）
FRE（"A"）

説　明 FRE関数は使用できるメモリのうち、使用していないメモリ（未使用領域）の大きさをバイト数で得ます。関数なので、他の命令と組み合せて使用します。

FRE（〈数値〉）は、プログラム（テキスト）領域、変数領域などのユーザーエリアの未使用領域の大きさをバイト数で得ます。〈数値〉はダミーの引数であればなんでもかまいません。

FRE（〈文字列〉）は、文字列領域の未使用領域の大きさをバイト数で得ます。文字列はダミーの引数です。

用　語

ユーザーエリア
パソコンに内蔵しているメモリ（RAM）は、パソコンが動作するために使うメモリ（ワークエリア）とプログラムを作るときに使うメモリ（ユーザエリア）、そして画面のデータを記憶するビデオメモリ（VRAM）からなります。ユーザーエリアは変数などの記憶用にも使われるため、多くの変数を使用すると、メモリの未使用領域は少なくなっていきます。

参　照 資料「メモリマップ」

GET

get（ゲット：得る）　　Disk BASIC

働　き ランダムアクセスファイルから、入出力用バッファにデータを1レコード読み込みます。

書き方 GET〔#〕〈ファイル番号〉〔,〈レコード番号〉〕

例　GET# 1，2

説　明 ディスケット内に記録しているランダムアクセスファイルから、入出力用バッファに1レコードのデータを読み込みます。

〈ファイル番号〉は、そのファイルをOPEN文で開いたときに指定した番号です。
〈レコード番号〉が省略されると、前のGET文、PUT文で使われたレコード番号の次のレコードが読み込まれます。

〈レコード番号〉として指定可能な最大の数は4294967295です。

注意！ GET文、PUT文はフロッピーディスクの説明書をよく読まれてからご使用ください。

GET DATE／TIME

get （ゲット：得る） MSX2

働　き　日付・時刻を得ます。

書き方　GET DATE〈文字変数〉〔,A〕
GET TIME〈文字変数〉〔,A〕

例

説　明　パソコン内の時計の日付・時刻を得ます。Aをつけるとアラームの日付・時刻です。日付・時刻のデータが〈文字変数〉に代入されます。アラーム機能については、各機種で異なりますので、パソコンの取扱説明書をご覧ください。またSET命令も参照してください。

GOSUB～RETURN

go to subroutine～return（ゴーサブ～リターン：サブルーチンへ行く，もどる）ステートメント

働　き　サブルーチンへ実行を移し、プログラムの実行後、もどります。

書き方　GOSUB〈行番号〉
RETURN〔〈行番号〉〕

例　GOSUB 100
RETURN 50

説　明　〈行番号〉から始まるプログラム（サブルーチン）へ実行を移し、プログラム実行後、RETURN文によりGOSUB文の次の文へ戻ります。

サブルーチンとは他から独立した1つのプログラムで、GOSUBで指定される行番号から、RETURN文までの命令のことです。
同じ計算を何度も行なうときなどサブルーチンにしておくと、GOSUB文によっていつでも実行できます。

RETURN文に〈行番号〉を指定すると、GOSUB文の次の文へは戻らず、指定した行に戻ります。

注意！
- サブルーチンの中から他のサブルーチンへ実行を移すこともできます（サブルーチンの多重化）。サブルーチンの多重化はメモリのスタック領域の容量が許す限り実行でき、メモリが不足すると“Out of memory”エラーとなります。
- サブルーチンは必ずGOSUB文によって実行しなければなりません。GOTO文などで実行中のプログラムにRETURN文があると“RETURN without GOSUB”エラーとなります。
- サブルーチンの中にCLEAR文があると“RETURN without GOSUB”エラーとなります。（スタック領域のクリアによって戻る行番号がわからなくなるためです）

参　照　RETURN，CLEAR，資料「メモリマップ」

GOTO

goto（ゴーツー：～へ行く）　　ステートメント

働　き 指定した行番号へ実行を移します。

書き方 GOTO 〈行番号〉

例　GOTO 50

説　明 〈行番号〉で指定した行へプログラムの実行を移します。
（無条件ジャンプといいます。）

GOTO文は “GO” と “TO” の間に１個のスペースを加えても同じ働きをします。

サンプルプログラム

20行～40行で無限ループを作ります。

```
10 '** ﾄﾒﾙ ﾄｷ ﾊ CTRL+STOPｷｰ **
20 PRINT A   ←┐
30 A=A+1      │ 40行を実行すると20行へもどる。
40 GOTO 20 ───┘
50 END
```

HEX$

hexa decimal $（ヘキサ ドル：16進数の文字列）　　関数

働　き 数値を16進数の文字列に変え、その結果を得ます。

書き方 HEX$（〈数値〉）

例　PRINT　HEX$(11)　　　実行結果……B

説　明 〈数値〉の値を16進数で表わす文字列に変えます。〈数値〉は－32768～32767の範囲です。
〈数値〉が負の場合、それに65336を加えた値が実際の〈数値〉として扱われます。

変換された結果の文字列は、ゼロサプレス（先頭の不要な0を取り除く）されています。

参　照 VAL，BIN$，OCT$

サンプルプログラム

```
10 PRINT " A    HEX$(A)"
20 FOR A=1 TO 16
30 A$=RIGHT$("     "+HEX$(A),5)
40 PRINT USING"##  @";A;A$
50 NEXT
```

IF～THEN…ELSE／IF～GOTO…ELSE

IF～THEN…ELSE（イフ ゼン エルス：もし～ならば…、そうでなければ）
IF～GOTO…ELSE（イフ ゴーツー エルス：もし～ならば…へジャンプする、そうでなければ）　ステートメント

働　き　条件判断をし、実行する命令を選びます。

書き方　① IF 〈条件〉 THEN ｛〈文〉〔：〈文〉…〕 または 〈行番号〉｝〔ELSE ｛〈文〉〔：〈文〉…〕 または 〈行番号〉｝〕

② IF 〈条件〉 GOTO 〈行番号〉〔ELSE ｛〈文〉〔：〈文〉…〕 または 〈行番号〉｝〕

例　① IF A>=B THEN PRINT A ELSE PRINT B
② IF A$="Y" GOTO 100 ELSE 200

説　明　〈条件〉が成立するならば、THEN文またはGOTO文以下の命令（ELSE文まで）を実行し、〈条件〉が成立しないならば、ELSE文以下が実行されます。
〈条件〉が成立しないときにELSE文がなければ、IF文の次の行が実行されます。

また、〈条件〉が成立している状態を“真”といい、〈条件〉は－1の値を持ちます。〈条件〉が成立しない状態を“偽”といい、〈条件〉が持つ値は0です。

THENのあとにGOTO文がくる場合、THENまたはGOTOの一方を省略することができます。また、ELSEのあとにGOTO文がくる場合も、GOTOを省略することができます。

参　照　資料「式と演算」

ちょっと一言

〈条件〉の種類
IF文で条件判断する〈条件〉には、次のものが使えます。

○数値や文字列の大小比較　例 IF A>255 THEN A=255
IF A$="Y" THEN 100 ELSE 200

○数値が0であるか否かの判断　例 IF A THEN 100
変数Aが0であれば、行番号100へジャンプします

○演算子を使った複合条件の判断　例 IF A<0 OR A>10 THEN 200
Aが負または10より大きいとき、行番号200へジャンプします

サンプルプログラム

```
10 COLOR 15,4,7:SCREEN 1
20 PRINT "U・・UP,D・・DOWN
30 PRINT "R・・RIGHT,L・・LEFT"
40 X=15:Y=10
50 LOCATE X,Y:PRINT "●";
60 A$=INKEY$
70 IF A$="U" THEN Y=Y-1
80 IF A$="D" THEN Y=Y+1
90 IF A$="R" THEN X=X+1
100 IF A$="L" THEN X=X-1
110 GOTO 50
```

G H I

INKEY$

in key $（インキードル：キー入力された文字）　関数

働き キーが押されていればその文字を、キーが押されていなければ" "（ヌルストリング）を得ます。

書き方 INKEY$

例　A$=INKEY$

説明 キーが押されている（キーボードのバッファが空でない）ときは、バッファの先頭の1文字を得ます。キーが押されていない（キーボードのバッファが空である）ときは、" "（ヌルストリング）を得ます。

なお、[CTRL] + [C]、[CTRL] + [STOP] のキー入力は、INKEY$関数で得ることはできません。

また、押されたキーは画面上に表示されず、INPUT命令のようにキー入力のところでプログラムの実行が停止することもありません。

このため、キー入力待ちをするためには、次のようなプログラムにします。

```
200 A$=INKEY$ : IF A$="" THEN 200
```

キー入力された文字を得る命令は、INPUT、LINE INPUT、INPUT$などもあります。

参照 INPUT、LINE INPUT、INPUT$

INP

input（インプット：入力）　関数

働き 入力ポートから1バイトのデータを得ます。

書き方 INP（<ポート番号>）

例　A=INP（&H90）　　実行結果……&H90の入力ポートから得たデータをAに代入する。

説明 <ポート番号> で指定された入力ポートから1バイト（8ビット）のデータを読み取ります。
<ポート番号> として指定できる値は0～255（&H0～&HFF）の範囲です。
関数のため、他の命令と組み合わせて使います。

参照 OUT，第5章「I/Oマップ」

input（インプット：入力する）　　ステートメント

働き キーボードからデータを入力し、指定した変数へ代入します。

書き方 INPUT〔"〈プロンプト文〉";〕〈変数名〉〔,〈変数名〉…〕

例　INPUT "なまえは"；NA$　　実行結果……「なまえは？」と表示される。

説明 キーボードからの入力を読み、変数に代入します。INPUT文を実行すると疑問符（？）と1つのスペースが画面に表示され、プログラムはキーボードからの入力待ちの状態となります。データは、リターンキーを押すことによって変数に代入されます。
"〈プロンプト文〉" を指定していれば、疑問符の前にプロンプト文が表示されてメッセージの役割をします。

〈変数名〉をカンマ（，）で区切って複数個指定した場合には、入力するデータもカンマで区切って〈変数名〉の数だけ入力します。

注意！
- グラフィックモードでINPUT文を使用すると自動的にテキストモードになります。
- 疑問符が表示された後、何も入力しないでリターンキーを押した場合には、変数の値は、INPUT文を実行する前と同じです。
- 文字変数に文字列を代入する場合、カンマや文字列の前後に意味のある空白を入力したいときには、引用符（"）で文字列を囲む必要があります。
- 〈変数名〉と入力するデータの型、数が一致しないときには次のメッセージが表示されます。

? Redo from start ……型が違っているとき。再び入力待ちとなります。
?? ……………………データの個数が足りないとき。再び入力待ちとなります。
? Extra ignored …………データの個数が多いとき。余分に入力されたデータが無視されます。

サンプルプログラム

- 色をカラーコードで指定し、カラーパターンを作ります。

```
10 '** ｶﾗｰ ﾊﾟﾀｰﾝ  **
20 INPUT "ｼｭｳﾍﾝ ﾉ COLOR･･";A
30 INPUT "ﾊｲｹｲ ﾉ COLOR･･･";B
40 INPUT "COLOR ﾉ ｶｽﾞ････";C
50 FOR F=1 TO C
60 PRINT F;
70 INPUT" ﾉ COLOR";A(F)
80 NEXT F
90 COLOR 15,B,A:SCREEN2
100 FOR T=1 TO C
110 LINE((T-1)*200/C+27,20)-(T*200/C+27,
171),A(T),BF
120 NEXT T
130 GOTO 130
```

INPUT＃

input＃（インプット シャープ：ファイルから入力する） ステートメント／Disk BASIC

働　き ファイルからデータを読み込みます。

書き方 INPUT＃〈ファイル番号〉,〈変数名〉〔,〈変数名〉…〕

例 INPUT＃1，A，B

説　明 指定したファイルからデータを読み込み、変数に代入します。指定するファイルは、あらかじめOPEN文によって開かれていなければなりません。

INPUT＃文で読み込むデータはPRINT＃文で出力したデータと対応しており、〈変数名〉の型はデータの型と対応する必要があります。

INPUT＃文はデータを読み込む対象がファイルであることを除けばINPUT文とほぼ同じです。

注意！
- ファイル中のデータが文字型の場合で、最初の文字が引用符（〃）のとき、そのデータは最初の引用符からつぎの引用符までの間に読み込まれた文字とみなされます。
- データの読み込み中にファイル中の読み込むデータがなくなったときは、〝Input past end〃エラーが表示されます。

参　照 OPEN，PRINT#，EOF

サンプルプログラム

カセットにデータを1度出力し、次に入力して表示します。

```
10 OPEN "CAS:FILES" FOR OUTPUT AS #1
20 READ A,B,C,D
30 PRINT #1,A,B,C,D
40 CLOSE:END
50 MAXFILES=2
60 OPEN "CAS:FILES" FOR INPUT AS #1
70 OPEN "CRT:FILES" FOR OUTPUT AS #2
80 INPUT #1,E,F,G,H
90 PRINT #2,E;F;G;H
100 CLOSE:END
110 DATA 1,2,3,4
```

```
run
Ok
cont
 1  2  3  4
Ok
```

カセットを巻き戻してください。

INPUT $

input $（インプット ドル：文字を入力する）　関数／Disk BASIC

働き 指定されたファイルから、指定された長さの文字を入力します。

書き方 INPUT $（〈文字数〉,〔#〕〈ファイル番号〉）

例　A$=INPUT$(4, #1)

説明 〈ファイル番号〉で指定されたファイルから〈文字数〉で指定した数の文字を入力します。指定するファイルはあらかじめOPEN文で開かれていなければはりません。

〈ファイル番号〉が省略された場合には、キーボードから入力が行なわれますが（このときはOPEN文でファイルを開く必要はありません）、INPUT文と異なり入力された文字は表示されません。

INPUT $文は[CTRL]+[STOP]を除くすべての文字をそのまま読み込みますので、INPUT文やLINE　INPUT文では入力することのできないリターンコードなどを入力することもできます。

注意！ INPUT$文は指定された〈文字数〉の文字が入力されるのを待ち続けますが、既にバッファに入力済みのデータがあるときは、バッファの文字を読み込みます。
（サンプルプログラムを実行したとき、[F6]キーを押してみてください。キーボードバッファが空になるまで（color 15, 4, 7の13文字分）"ダメ"を表示します。）

参照 INPUT, LINE　INPUT

サンプルプログラム "MSX"という文字を入力するまで"ダメ！"を表示します。

```
10 'ﾊﾟｽﾜｰﾄﾞ･･･MSX
20 P$="MSX"
30 PRINT "ﾊﾟｽﾜｰﾄﾞ:";
40 A$=INPUT$(3)
50 IF A$=P$ GOTO 80
60 PRINT "ﾀﾞﾒ!"
70 BEEP:GOTO 30
80 PRINT "Ok!"
90 END
```

```
run
ﾊﾟｽﾜｰﾄﾞ:ﾀﾞﾒ!
ﾊﾟｽﾜｰﾄﾞ:Ok!
Ok
```

I

INSTR

in string（イン ストリング：文字列の中）

働　き 文字列の中から指定する文字列を捜します。

書き方 INSTR（〔〈数〉,〕〈文字列１〉,〈文字列２〉）

例　L＝INSTR（A＄,“END”）

説　明 〈文字列１〉の中から〈文字列２〉を捜し、見つかれば先頭文字から何文字目にあるのかを、見つからなければ0を得ます。関数のため、他の命令と組み合せて使用します。

〈数〉は捜し始める位置を0～255の整数で指定します。省略したときは、〈文字列１〉の左側から捜します。

〈文字列２〉の長さが〈文字列１〉より大きいとき、および〈文字列１〉がヌルストリング（“ ”）のとき、〈数〉が〈文字列１〉の長さより大きいときは、0を得ます。
〈文字列２〉がヌルストリングのときは、〈数〉の値を得ます。（省略しているときは1です。）

参　照 LEN

サンプルプログラム

キー入力されたA～Zの文字が、アルファベットの何文字かを表示します。

```
10 A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
20 I$=INKEY$:IF I$="" GOTO 20
30 I=INSTR(A$,I$)
40 IF I=0 THEN PRINT "ｱﾙﾌｧﾍﾞｯﾄ ﾃﾞﾅｲ" ELS
E PRINT I$;" ﾊ ";I;"ﾊﾞﾝﾒ ﾃﾞｽ"
50 GOTO 20
```

INT

integer（インテジャー：整数）　　関数

働　き 整数値を得ます。

書き方 INT（〈数〉）

例　PRINT INT（3.1416）　　実行結果……3

説　明 〈数〉の値を越えない最大の整数を得ます。
FIX関数との違いはサンプルプログラムをご覧ください。

参　照 FIX，CINT

サンプルプログラム

INT関数とFIX関数の比較

```
10 PRINT "  X   INT(X) FIX(X)"
20 PRINT
30 FOR K=-3.14 TO 3.86
40 A=INT(K):B=FIX(K)
50 PRINT USING"##.##  ###  ###";K,A,B
60 NEXT
```

```
run
  X    INT(X)  FIX(X)

-3.14    -4     -3
-2.14    -3     -2
-1.14    -2     -1
-0.14    -1      0
 0.86     0      0
 1.86     1      1
 2.86     2      2
 3.86     3      3
Ok
```

INTERVAL ON／OFF／STOP

interval on/off/stop（インターバル オン/オフ/ストップ：タイマー割込みの許可/禁止/保留） ステートメント

働き タイマによる割込みを許可、禁止、保留します。

書き方
① INTERVAL　ON
② INTERVAL　OFF
③ INTERVAL　STOP

説明 ON INTERVAL　GOSUB文で設定したタイマ割込みを許可するか（ON)、禁止するか（OFF)、保留するか（STOP）を指定します。

①INTERVAL　ON文はタイマ割込みを許可します。
この命令を実行しておくと、ON INTERVAL　GOSUB文で設定した時間ごとに割込みが発生し、実行中のプログラムを中断してタイマ割込み処理ルーチンが実行されます。
タイマの時間チェックは、プログラム中の個々の命令文を実行するたびに行なわれます。

②INTERVAL　OFF文はタイマ割込みを禁止します。
ON INTERVAL　GOSUB文で設定した時間が経過しても割込みは発生しません。また、タイマの時間チェックも行ないません。

③INTERVAL　STOP文はタイマ割込みを保留します。
この命令を実行すると、割込みの発生は禁止されますが、タイマの時間チェックは行なわれ記憶されます。INTERVAL　STOP文の後、INTERVAL　ON文を実行するまでにタイマ割込みの時間が経過していると、INTERVAL　ON文の実行直後に割込みが発生します。

用語 **割込みと割込み処理ルーチン**
GOTO文の無条件ジャンプと異なり、ある条件（時間など）が整った時点で指定されたプログラム（処理ルーチン）へ実行を移すことを割込みといいます。
割込みが発生したときに、その条件などに応じて処理を行なうプログラムを処理ルーチンといいます。

参照 ON INTERVAL GOSUB，第2章

サンプルプログラム

時間を秒で表示し、5秒ごとにブザーを鳴らします。

```
10  TIME=0 :ON INTERVAL=300 GOSUB 60
20  INTERVAL ON
30  LOCATE 10,10
40  PRINT USING "###.##";TIME/60
50  GOTO 30
60  BEEP:RETURN
```

KEY

key（キー：キーボードのキー）　コマンド

働き　ファンクションキーの内容を決めます。

書き方　KEY 〈キー番号〉,〈文字列〉

例　KEY 8,"PRINT"+CHR$(34)

説明　〈キー番号〉で指定するファンクションキー（キーボードの上部）の内容を、〈文字列〉で定義します。〈キー番号〉は1～10の整数で、F1～F10のキーに対応しています。

〈文字列〉は最大15文字までの文字およびコントロールコードです。16文字以上定義しようとしても、残りは無視されます。

注意！
- リターンキーや「"」はCHR$関数を使って定義します。
  リターンキー……コントロールコード13
  「"」……コントロールコード34
- 一度定義した内容は、再び設定し直すかパソコンをリセットするまで変わりません。

ちょっと一言

プログラムを入力するときには、次のような文字列を定義しておくと便利です。
"PRINT"+CHR$(34)　,"FOR "　,"NEXT"
"SCREEN "　,"DATA "

KEY LIST

key list（キーリスト：キーの一覧表）　コマンド

働き　ファンクションキーの定義内容を画面に表示します。

書き方　KEY LIST

説明　ファンクションキーに定義されている〈文字列〉をF1からF10まで順に、画面に表示します。

通常（KEY ONの状態）、ファンクションキーの内容はテキスト画面の最下行に表示されていますが、SCREEN 1では5文字まで、SCREEN 0でも7文字までしか表示されません。このため、ファンクションキーの内容を確認するときなどにKEY LIST文を実行します。MSX2のSCREEN 0，WIDTH80では15文字までが表示されます。

F1……"color "　　F6……"color 15, 4, 7"+CHR$(13)
F2……"auto "　　F7……"cload"+CHR$(34)
F3……"goto "　　F8……"cont"+CHR$(13)
F4……"list "　　F9……"list"+CHR$(13)+CHR$(30)+CHR$(30)
F5……"run"+CHR$(13)　　F10……CHR$(12)+"run"+CHR$(13)

KEY(n) ON／OFF／STOP

key(n) on/off/stop (キー(番号) オン/オフ/ストップ：ファンクションキー(n)の割込みを許可/禁止/保留) ステートメント

働き ファンクションキーによる割込みを許可、禁止、保留します。

書き方 ① KEY (《キー番号》) ON
② KEY (《キー番号》) OFF
③ KEY (《キー番号》) STOP

例 KEY (1) ON

説明 ファンクションキーが押されたときに発生する割込みを許可するか (ON)、禁止するか (OFF)、保留するか (STOP) を指定します。

割込みはON KEY GOSUB文によって設定し、〈キー番号〉はファンクションキー[F1]～[F10]の番号を表します。

①KEY(n)ONは割込みを許可します。この命令を実行すると指定した〈キー番号〉のファンクションキーを押すごとにON KEY GOSUB文によって設定された割込み処理ルーチンにジャンプします。KEY(n)ON状態では、ファンクションキーを押してもキーの内容は画面に表示されません。

②KEY(n)OFFは割込みを禁止します。この命令を実行すると、ファンクションキーを押しても割込み処理ルーチンへジャンプしません。

③KEY(n)STOPは割込みを保留します。この命令を実行すると割込みの発生は禁止されますが、ファンクションキーが押されたかチェックは行ないます。KEY(n)STOP文の後、KEY(n)ON文を実行するまでにファンクションキーを押していると、KEY(n)ON文の実行直後に割込みが発生します。

参照 ON KEY GOSUB

KEY ON／OFF

key on/off (キーオン/オフ：キー内容を表示する/表示しない) ステートメント

働き ファンクションキーの定義内容を画面に表示するか、表示しないかを指定します。

書き方 ① KEY ON

② KEY OFF

説明 ①KEY ONは、テキスト画面の最下行にファンクションキーの定義内容を表示します。(パソコンのリセット時はKEY ON状態です)
ファンクションキーの定義内容 (通常は[F1]～[F5]、シフトキーを押すと[F6]～[F10]) を表示しているときは、画面の最下行は使用できません。

②KEY OFFは、ファンクションキーの表示を消します。このとき、画面の最下行も使用可能となります。KEY OFFの状態でも、ファンクションキーからの入力はできます。

参照 KEY

KILL

kill（キル：つぶす，とる）　Disk BASIC

働　き　ディスケットからファイルを消去します。

書き方　KILL “<ファイルスペック>”

例　KILL “DEMO”

説　明　<ファイルスペック> で指定したファイルを消去します。
<ファイルスペック> の中で、ディスクのドライブ番号が指定されていないときは、現在使用されているディスクドライブとみなされます。なお、ファイル名を省略することはできません。

開いているファイルに対してKILL文を実行しますと、“File still open” エラーとなります。CLOSE文を実行してからKILL文を実行してください。

LEFT$

left $（レフト ドル：左側の文字列）　関数

働　き　文字列の左側から指定した長さの文字列を得ます。

書き方　LEFT$（<文字列>,<数>）

例　A$=“ABCD”：D$=LEFT$(A$，2)　　実行結果……D$は “AB” です。

説　明　<文字列> の左から <数> で指定した文字数の文字列を得ます。
<数> の範囲は0～255です。関数なので、他の命令と組み合せて使います。

<数> が <文字列> の文字数より大きいときは <文字列> のすべてを、<数> が0のときはヌルストリングを得ます。

参　照　MID$，RIGHT$

サンプルプログラム

```
10 ' ** LEFT$ **
20 A$="LEFT$ハ モジレツ ノ ヒダリカラ トリダシマス"
30 L=LEN(A$)
40 FOR R=1 TO L
50 B$=LEFT$(A$,R)
60 PRINT B$
70 NEXT
```

LEN

length（レングス：長さ）　関数

働　き　文字列の文字数を得ます。

書き方　LEN（〈文字列〉）

例　A$="ABCD"：L=LEN（A$）　　実行結果……Lは4です。

説　明　〈文字列〉の総文字数を得ます。関数なので他の命令と組み合せて使用します。

〈文字列〉には、画面に表示されない文字（スペースやコントロールコード1～31）も含みます。

サンプルプログラム

MSX BASICの文字数（9）を表示します。

```
10 A$="MSX BASIC"
20 L=LEN(A$)
30 PRINT L
```

LET

let（レット：～させる）　ステートメント

働　き　変数に式を代入します。

書き方　〔LET〕〈変数名〉=〈式〉

例　LET　A=7

説　明　〈変数名〉に〈式〉を代入します。等号（=）だけでも代入を表わすので、LETは省略でき普通は使いません。

〈式〉は文字列の場合もありますが、〈変数名〉の型と〈式〉の型は一致しなくてはなりません。

サンプルプログラム

```
10 LET A=2
20 B=7
30 PRINT A
40 PRINT B
50 END
```

K
L

LINE

line（ライン:線）　　ステートメント

働　き　画面上に直線または四角形を描きます。

書き方　LINE〔 (x1,y1) ｜ STEP(x1,y1) 〕－ (x2,y2) ｜ STEP(x2,y2) 〔,〈カラーコード〉〕〔, B ｜ BF 〕〔,〈論理演算子〉〕

（〈論理演算子〉は MSX2のみ）

例　LINE (0,0)－(117,100),15

説　明　指定した2点、(x1, y1) と (x2, y2) を結ぶ直線または四角を描きます。(x1, y1)が始点、(x2, y2) が終点です。始点 (x1, y1) を省略した場合LP（最終参照点）を始点とします。
また、座標の指定にSTEPを付けると、LPからの相対座標の指定になります。

〈カラーコード〉は直線や四角形の色を指定します。〈カラーコード〉を省略すると、現在COLOR文によって設定されている前景色となります。

Bを指定すると(x1, y1) と (x2, y2) の2点を対角とする四角形を描きます。BFを指定すると (x1, y1) と (x2, y2)点を対角とする四角形を描き、その内側を塗りつぶします。
BはBox、BFは、Box Fillの意味です。なお、B,BFを指定しないときは(x1, y1)と(x2, y2)を結ぶ直線を描きます。

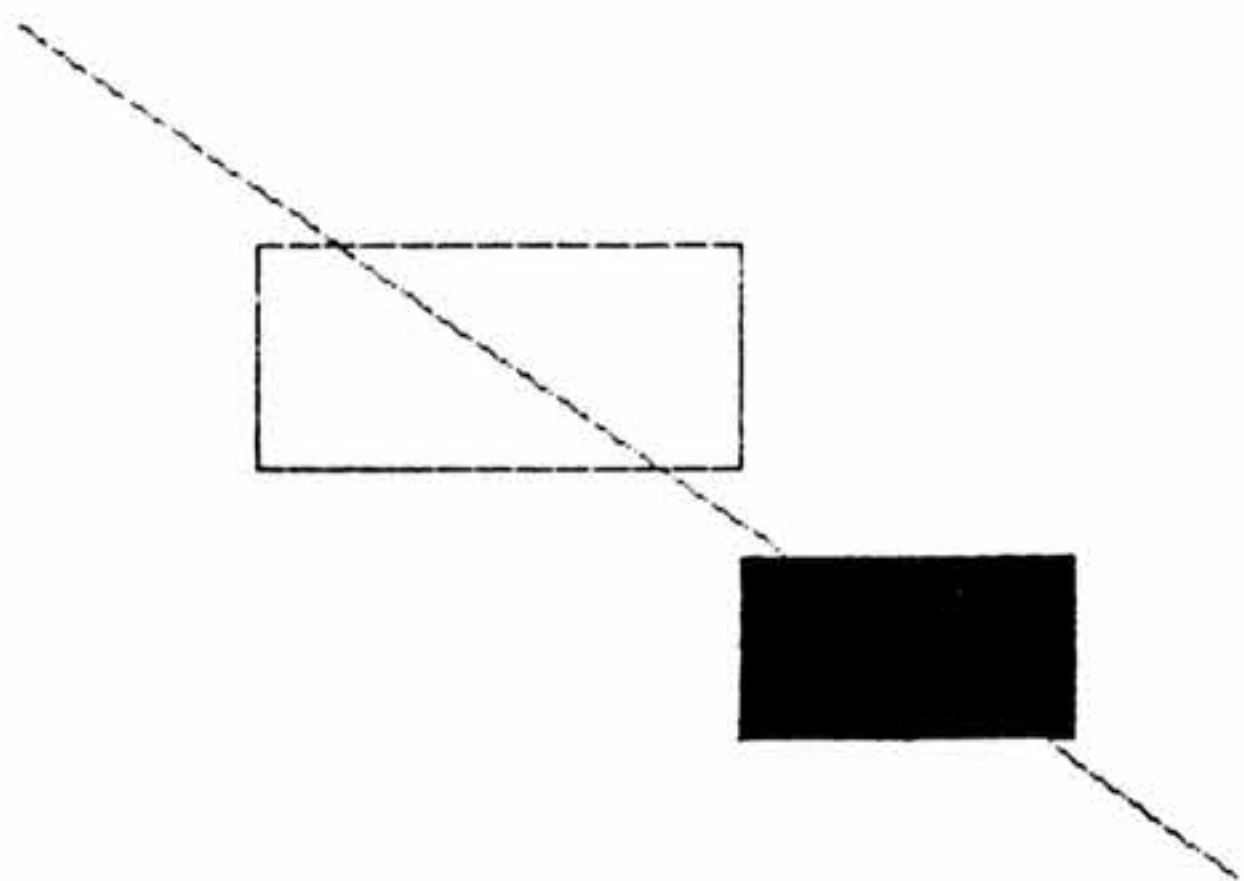
LINE文はグラフィックモードのみで使用し、テキストモードで使用すると、"Illegal function call" エラーとなります。
また、LINE文を実行するとLPは (x2, y2) に移動します。

MSX2では、〈論理演算子〉を指定することによって、他のグラフィック命令で描いた図と重なるLINEの線の色を以下のように設定できます。なにも指定しないときは、PSETと同じです。指定色をCO、LINEの下にある前景色をSとすると、

〈論理演算子〉	表示色	例（CO＝&B1001明るい赤、S＝&B1010黄）
AND	CO AND S	&B1000…赤
OR	CO OR S	&B1011…明るい黄
XOR	CO XOR S	&B0011…明るい緑
PSET	CO	&B1001…明るい赤
PRESET	NOT(CO)	&B0110…暗い赤

この指定は、SCREEN 5～8 に限られます。なお、SCREEN 7と8 はVRAM 128 KBの機種のみ使用できます。

サンプルプログラム

```
10 SCREEN 5
20 LINE(0,0)-(255,191),12
30 LINE(50,50)-(150,100),9,B
40 LINE(150,120)-(220,160),10,BF
50 GOTO 50
```

LINE INPUT

line input （ラインインプット：1行の入力） ステートメント

働　き キーボードから文字列を1行分入力し、文字変数へ代入。

書き方 LINE INPUT〔"<プロンプト文>"；〕<文字変数>

例 LINE INPUT"DATE"；D$

説　明 キーボードから入力された全ての文字データを<文字変数>に代入します。LINE INPUT文を実行すると、パソコンはキーボードからの入力待ちの状態となります。以降、リターンが押されるまでに入力された文字データ（254文字以内）が<文字変数>に代入されます。

LINE INPUT文はINPUT文と異なり、カンマ(，)によってデータが区切られることはありません。また、疑問符(?)は表示されません。
"<プロンプト文>"を指定していれば入力待ちの状態となる前にプロンプト文を表示します。

LINE INPUT文の実行は CTRL + C または CTRL + STOP キーの入力によって中断することができます。この後、CONT命令により実行を再開すると、中断したLINE INPUT文の最初から実行されます。

参　照 LINE INPUT#，INPUT

サンプルプログラム

```
10 ' ** LINE INPUT **
20 INPUT"      INPUT :";A$
30 LINE INPUT"LINE INPUT :";B$
40 PRINT
50 PRINT"INPUT : ";A$
60 PRINT"LINE INPUT ﾃﾞﾊ ";CHR$(34);CHR$(
44);" ｦ ﾆｭｳﾘｮｸﾃﾞｷﾏｽ｡"
70 PRINT B$
```

L

LINE INPUT ＃

line input ＃（ラインインプットシャープ：1行のデータ入力）　ステートメント

働き　1行分のデータをファイルから文字変数へ読み込みます。

書き方　LINE INPUT　＃〈ファイル番号〉，〈文字変数〉

例　LINE INPUT　#1，A$

説明　あらかじめOPEN文でINPUTモードを指定して開いておいたファイルから1行単位（254文字以内）のデータを読み込み文字変数へ代入します。
ここでの1行とは、改行コード〔キャラクタコード13(&H0D)とキャラクターコード10(&H0A）が連続したもの〕を入力するまでのデータ入力コードです。

SAVE文で作成したアスキー形式のプログラムファイルには、一行一行が改行コードで区切られて書き出されています。また、PRINT＃文で作成したファイルにはPRINT＃文を実行するごとに数値や文字列の並びが改行コードで区切られて書き出されています。LINE INPUT＃文ではこれらのファイルから改行コードにはさまれているデータを読み込むときなどに使います。

データ中にキャラクタコードCHR$(13)＋キャラクタコードCHR$(10)が含まれている場合、改行コードとみなされますが、CHR$(10)＋CHR$(13)の組み合わせは改行コードとはみなされません。

〈ファイル番号〉はOPEN文で指定した番号です。

なお、改行コードではさまれているデータが254文字以上であれば、254文字までが文字変数に代入されます。

参照　OPEN、CLOSE、PRINT＃、第1章「ファイル」

サンプルプログラム

```
10 ' ** LINE INPUT# **
20 OPEN "CAS:demo" FOR OUTPUT AS #1
30 A$="MSX BASIC"
40 PRINT #1,A$
50 CLOSE:END
60 MAXFILES=2
70 OPEN "CAS:demo" FOR INPUT AS #1
80 OPEN "CRT:" FOR OUTPUT AS #2
90 LINE INPUT #1,A$
100 PRINT #2,A$
110 CLOSE
120 END
```

LIST/LLIST

list　（リスト：リストアップする）
Llist　（ラインリスト：プリンタにリストアウトする）　**コマンド**

働き プログラムの全部または一部を画面やプリンタに出力します。

書き方 LIST〔〈行番号〉〕〔－〈行番号〉〕
LLIST〔〈行番号〉〕〔－〈行番号〉〕

例 LIST 100－200
LLIST

説明 メモリの上にあるプログラムをLISTは画面に表示し、LLISTはプリンタに出力します。(リストアウトします。)

LIST
〈行番号〉を省略した場合は、プログラムの先頭から最後まで全部がリストアウトされます。

LIST100
〈行番号〉を指定すると、その行だけがリストアウトされます。

LIST100－
〈行番号〉－を指定すると、指定した行番号以降のプログラムがすべてリストアウトされます。

LIST100－200
〈行番号〉－〈行番号〉を指定すると、1番目に指定した行番号から2番目に指定した行番号までのプログラムがリストアウトされます。

LIST－200
－〈行番号〉を指定すると、プログラムの先頭から指定した行番号までのプログラムがリストアウトされます。

LIST.
〈行番号〉の代りに〝.〃(ピリオド)を指定するとBASICが現在注目している行(スクリーンエディタやLIST文などで最後に参照した行)を指します。

リストアウトの途中で［STOP］キーを押すとリストアウトは中断し、再び［STOP］キーを押すと、リストアウトを再開します。また、リストアウトの途中で［CTRL］+［STOP］キーを押すと、リストアウトは終了します。

L

LOAD

load （ロード：積む、詰め込む）　　コマンド/Disk BASIC

働　き　プログラムをファイルから読み込みます。（ロードします）

書き方　LOAD "<ファイルスペック>" 〔,R〕

例　LOAD　"CAS:TEST"

説　明　<ファイルスペック>で指定したプログラムをメモリにロードします。ロードするプログラムはSAVE文を使ってセーブ（記録）されたものです。LOAD文を実行すると、LOAD文実行前にメモリにあったプログラムは消去されます。また、すべての開いているファイルが閉じられ、変数の値は初期化されます。

Rオプションをつけると、ファイルは開いたままでプログラムをロード後、ただちに実行を開始します。Rオプションをつけないと、ロードだけが行なわれます。

LOADは、指定されたファイルを見つけてプログラムのロードを開始するまでは、メモリのプログラムを保存します。
LOAD命令では、CSAVEでセーブしたプログラムをロードすることはできません。

注意！　<ファイルスペック>により、プログラムをロードする装置が異なります。

LOAD "CAS:TEST"―― カセットにセーブされているプログラムをロードします。
LOAD "TEST"――――― フロッピーディスクが接続されているときは、フロッピーディスクからロードし、フロッピーディスクが接続されていないときは、カセットテープからロードします。

参　照　SAVE、MERGE、CLOAD

LOC

locate （ロケイト：位置）　　Disk BASIC

働　き　ファイル中の現在の位置を得ます。

書き方　LOC（<ファイル番号>）

例　IF LOC（1）>50 THEN STOP

説　明　ランダムファイルを指定した時は、LOC関数は、最後に読んだり書いたりしたレコード番号を値として得ます。ただし、OPEN命令の実行直後はLOC関数の値は0になっています。

シーケンシャルファイルを指定した時は、そのファイルがオープンされてから読み出されたり書き込まれたレコード数を得ます。ファイルがシーケンシャル入力モードでオープンされた場合は、BASICは最初のセクタを読むので、そのファイルを読む前でもLOC関数の値は1となります。

関数ですので、他の命令と組み合せて使います。

参　照　EOF,

LOCATE

locate （ロケイト:置く、設定する） ステートメント

働　き テキスト画面のカーソルの位置および働きを指定します。

書き方 LOCATE〔〈X座標〉〕〔,〈Y座標〉〕〔,〈カーソルスイッチ〉〕

例 LOCATE15, 10

説　明 カーソルをテキスト画面の指定した位置（X, Y）へ移動します。

この命令は、PRINT文で表示する文字やINPUT文で表示するプロンプト文の開始位置を指定するときなどに用います。（入力待ちのときの?の位置を指定できます。）

〈X座標〉は画面の右端を0とする水平座標を表す数で、32×24テキストモードのとき0～31、40×24テキストモードのとき0～39です。省略した場合は0とみなされます。

MSX2の80×24テキストモードのとき0～79です。

〈Y座標〉は画面の上端を0とする垂直座標を表す数で、0～22（KEY OFFで0～23）です。省略した場合は現在カーソルがセットされている行とみなされます。

〈カーソルスイッチ〉は画面上のカーソルの表示状態を決めるスイッチです。

0：カーソルはキー入力待ちのとき以外表示されない

1：カーソルは常に表示される

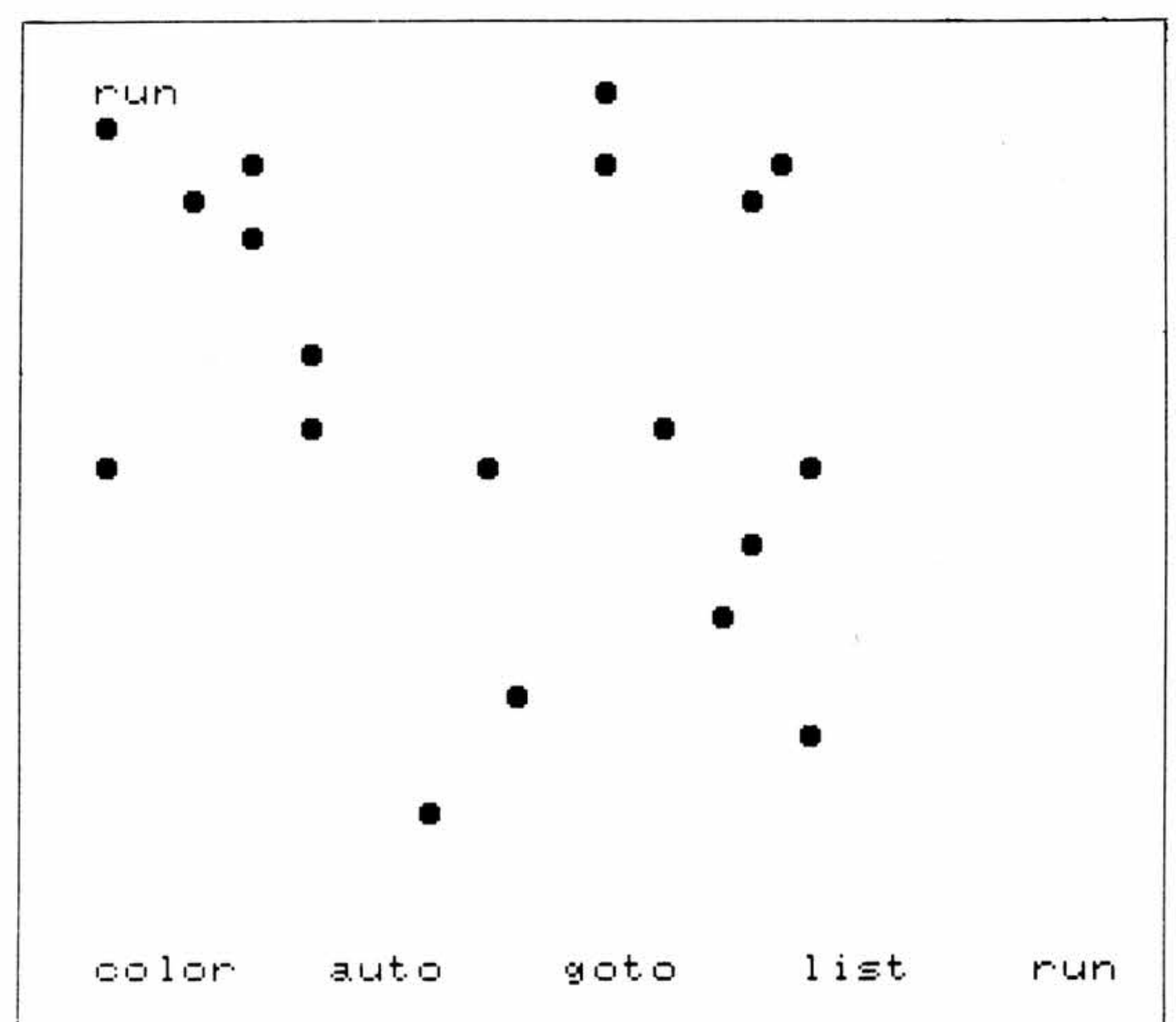
LOCATE文はテキストモードでのみ使用することができます。

サンプルプログラム

適当な位置に“●”を描きます。（止めるときは CTRL キーを押しながら STOP キーを押す）。

```
10 ' ** LOCATE  **
20 X=RND(1)*30
30 Y=RND(1)*21
40 LOCATE X,Y:PRINT "●"
50 GOTO 20
```

run

color　auto　goto　list　run

LOF

length of file (レングス オブ ファイル：ファイルの長さ)　Disk BASIC

働　き　指定されたファイルの大きさを得ます。

書き方　LOF (〈ファイル番号〉)

例　REC=LOF (2)

説　明　指定されたファイルの実際の大きさをバイト単位で得ます。

指定されたファイルがランダムファイルであった場合は、そのファイルの最大レコード番号にレコード長をかけたものに相当します。

LOG

logarithm (ログ：対数)　関数

働　き　自然対数を得ます。

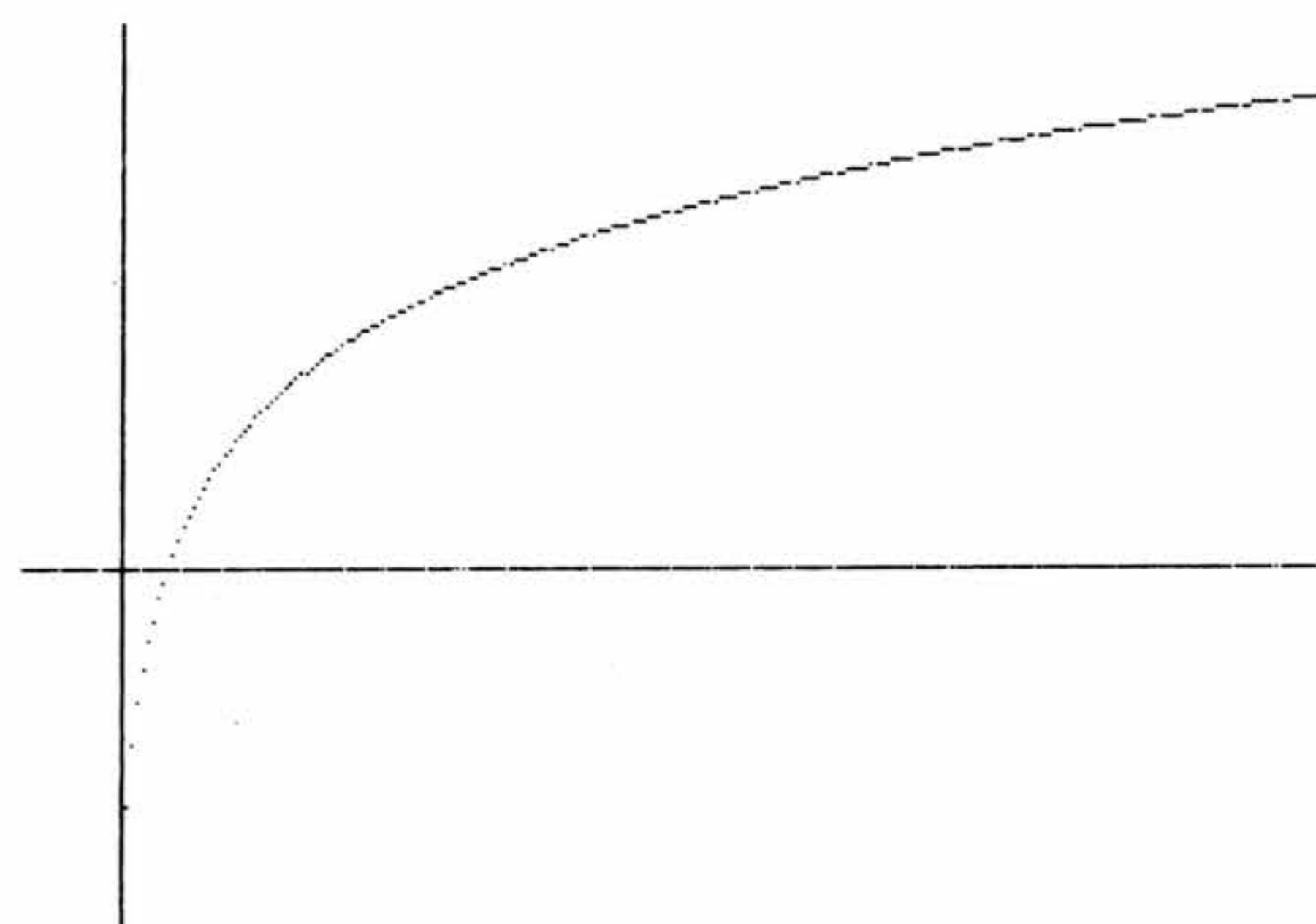
書き方　LOG (〈数〉)

例　PRINT LOG (A)

説　明　〈数〉で指定した値の自然対数 (eを底とした対数) を求めます。〈数〉は0より大きくなければなりません。

〈数〉の型はどのような型でもかまいませんが、結果は倍精度実数で得られます。また、関数のため、他の命令と組み合せて使います。

サンプルプログラム　自然対数曲線

```
10 ' ** LOG **
20 SCREEN 2
30 LINE(0,115)-(255,115)
40 LINE(20,0)-(20,191)
50 FOR L=.1 TO 25 STEP .1
60 Y=LOG(L)/3.2*100
70 PSET(20+X,115-Y)
80 X=X+1
90 NEXT
100 GOTO 100
```

LPOS

line position（ライン ポジション：プリンタの位置）　システム変数

働　き　プリンタヘッドの現在位置を得ます。

書き方　LPOS（〈式〉）

　例　L＝LPOS（0）

説　明　得られる値はプリンタバッファ上のプリンタヘッドの水平方向の位置で、必ずしも物理的なプリンタヘッドの位置とは限りません。

〈式〉はダミーの引数で意味を持ちません。変数か定数であれば何であってもかまいません。

参　照　WIDTH, LPRINT

LPRINT/LPRINT USING

line print（ラインプリント：プリンタへの出力）
line print using（ラインプリントユージング：プリンタへの出力）　ステートメント

働　き　文字列や数値を指定した書式でプリンタに出力します。

書き方　LPRINT〔〈式〉・・・〕
LPRINT USING〈書式〉;〈式〉・・・

　例　LPRINT A$, B
LPRINT USING "@####. ##";A$;B

説　明　〈式〉で表す文字列や数値をプリンタに出力します。複数個の〈式〉を指定する場合は〈式〉の間をカンマ、またはセミコロンで区切ります。LPRINT文、LPRINT USING文は、出力先がプリンタであることを除けば、PRINT 文や PRINT USING 文とまったく同じです。

注意！　L?と入力してもLPRINTとしては使えません。

参　照　PRINT, PRINT USING

LSET/RSET

left set　（レフトセット：左にそろえる）
right set　（ライトセット：右にそろえる）　**Disk BASIC**

働き ランダム入出力用バッファにデータを転送する。（PUT命令のための準備）

書き方 LSET　〈文字型変数〉＝〈文字式〉
RSET　〈文字型変数〉＝〈文字式〉

例　LSET　D$=MKI$（D）

説明 〈文字式〉の文字が、FIELD文で指定された文字より短い場合は、LSET文においては左詰め、RSET文においては右詰めでフィールド内を埋め、余分な所は空白で満たされます。
逆にFIELD文での割り当てより長い場合は、LSET文、RSET文両方とも文字列の右側が失われます。
数値のデータを扱うときは、あらかじめそれらをMKI$，MKS$，MKD$を使って文字型データに変換しておかなければなりません。

参照 FIELD, PUT, MKI$/MKS$/MKD$,　CVI/CVS/CVD

MAXFILES

maxfiles（マックスファイルズ：ファイルの最大数）　**ステートメント**

働き 同時に使用するファイルの数を定義します。

書き方 MAXFILES＝〈ファイル数〉

例　MAXFILES＝5

説明 OPEN文で開いて使用するファイルの数を定義します。以降、指定した数のファイルを同時に開いて使用することができます。

〈ファイル数〉は0〜15の範囲です。ファイル1個について267バイトのファイルコントロールブロック領域が確保されますので、必要なファイル数だけを指定してください。

MAXFILES文を実行すると、変数がすべて初期化され、現在オープンしているファイルはクローズされます。また、以前にDEF文で定義した情報も無効となります。

注意！ MAXFILES＝0とすると、ファイルの入出力はSAVEとLOADのみが有効となります。
また、MAXFILES文を実行する前に使用できるファイル数は1です。

MERGE

merge（マージ：合わせる）　コマンド/Disk BASIC

働　き アスキーファイルされているプログラムをメモリにあるプログラムに混合（マージ）します。

書き方 MERGE "〈ファイルスペック〉"

例 MERGE "CAS:DEMO"

説　明 メモリにあるベーシックのプログラムに〈ファイルスペック〉で指定したファイルのプログラムを混合して1つのプログラムにし、メモリ上に置きます。

〈ファイルスペック〉で指定するファイルはSAVE文を使ってアスキー形式でセーブされていなければなりません。そうでない場合には、エラーとなります。

カセットテープからプログラムを読み込むときは〈ファイルスペック〉のデバイス名には"CAS:"を指定します。また、〈ファイルスペック〉のファイル名を省略すると、カセットテープの位置している場所以降にある最初のアスキーファイルがロードされます。

フロッピーディスクからプログラムを読み込むときは、〈ファイルスペック〉のファイル名を省略することはできません。

ファイル中のプログラムと、メモリ中のプログラムに同一の行番号があった場合には、ファイル中の行でメモリ中の行を置き換えます。

プログラム中でMERGEコマンドを実行すると、終了後はダイレクトモードに戻ります。

参　照 SAVE，LOAD

M

L

MID$

middle$ （ミドルドル：文字列の中） 関数

働 き 文字列の中から指定した長さの文字列をとり出します。

書き方 MID$（〈文字列〉，〈式1〉〔，〈式2〉〕）

例 PRINT MID$（NAME，3，2）

説 明 〈文字列〉の左から〈式1〉番目の文字から〈式2〉文字の文字列をとり出します。〈式2〉の値は0から255の範囲、また、〈式1〉の値は1から255の範囲です。

〈式2〉を省略した場合、および〈文字列〉の〈式1〉番目の文字から右の文字数が〈式2〉より小さい場合は、〈文字列〉の〈式1〉番目より右すべての文字列をとり出します。

〈文字列〉の文字数が〈式1〉より小さければ、とり出す文字列は〝〟（ヌルストリング）となります。
関数のため、他の命令と組み合せて使います。

参 照 RIGHT$，LEFT$，MID$（ステートメント）

サンプルプログラム 日付の表示

```
10 D$="85/01/01"                              ← 日付を設定
20 A$=LEFT$(D$,2)                             ← 年をもとめる
30 B$=MID$(D$,4,2)                            ← 月をもとめる
40 C$=RIGHT$(D$,2)                            ← 日をもとめる
50 IF LEFT$(B$,1)="0"THEN MID$(B$,1)=" "      ← 0を空白とする
60 IF LEFT$(C$,1)="0"THEN MID$(C$,1)=" "      ←
70 PRINT A$+"年"+B$+"月"+C$+"日"              ← 表示する
```

MID$

middle $ （ミドルドル：文字列の中） ステートメント

働 き 文字列の一部を他の文字列で置き換えます。

書き方 MID$（〈文字列1〉，〈式1〉〔，〈式2〉〕）＝〈文字列2〉

例 MID$（A$，3）＝〝ABC〟 実行結果……A$の3文字目からが〝ABC〟となります。

説 明 〈文字列1〉の〈式1〉番目から〈式2〉個の文字 、〈文字列2〉の文字列で置き換えます。

〈文字列1〉、〈文字列2〉は文字列または文字変数です。また、〈式1〉、〈式2〉は数式または数値です。

〈式1〉の値は〈文字列1〉の文字数より大きくてはいけないし、0以下であってもいけません。

〈式2〉を省略した場合，または〈文字列2〉の文字数より多く指定した場合は、〈文字列2〉のすべての文字と置き換わります。

参 照 MID$関数

MKI$/MKS$/MKD$

MKI$(インテジャードル)/MKS$(シングルドル)/MKD$(ダブルドル)　　Disk BASIC

働　き 各数値を内部表現に対応したキャラクターコードに変換します。

書き方 MKI$　(〈整数表記〉)
MKS$　(〈単精度表記〉)
MKD$　(〈倍精度表記〉)

説　明 これらの関数は、数値をランダム入出力用バッファに対してLSET／RSETで書き込むときに使用します。MKI$は整数値を2バイトの文字列に、MKS$は単精度数値を4バイトの文字列に、そしてMKD$は倍精度数値を8バイトの文字列にそれぞれ変換する。

数値から文字への変換は数値が持つ内部表現（2進数表現）の値をそのままそれに対応するキャラクタコードにすることによって行われます。この逆の動作をする関数としてはCVI／CVS／CVD関数が用意されています。

参　考 実際の文字列と変換された数値との関係は、MKI$を例に説明すると次のようになります。
A$=MKI$（A%）として、A%=16710 の場合には、
16710=&H4146=CHR$(&H41)+CHR$(46)="AF"(実際には"FA"で記録されます。)
となります。つまり、数値データをキャラクターコードとして記録します。
MKS$, MKD$の場合は、各キャラクターコードを正規化された浮動小数点形式の数値データとみなすことになります。
例
MKS$(2.34567E+40)=CHR$(&H69)+CHR$(&H23)+CHR$(&H45)
+CHR$(&H67)
="i#Eg"

指数

〈数〉E−65……&H0	−〈数〉E−65……&H80
〈数〉E±0……&H41	−〈数〉E±0……&HC1
〈数〉E+65……&H7F	−〈数〉E+62……&HFF

参　照 CVI／CVS／CVD, LSET／RSET

MOTOR ON/OFF

motor on/off (モータオン/オフ)　　ステートメント

働　き カセットテープレコーダのモータのON/OFFをコントロールします。

書き方 MOTOR〔ON | OFF〕

例　MOTOR

説　明 リモート端子のあるカセットレコーダに対して有効です。
ONを指定すればモータはONとなり、OFFを指定すればモータはOFFとなります。また、ON/OFFを指定しない場合、現在モータがONの状態ならばOFFに、OFFの状態ならばONにします。

NAME

name（ネーム：名前）　　Disk BASIC

働　き ファイルの名前を変更します。

書き方 NAME "〈現在のファイル名〉" AS "〈新しいファイル名〉"

例 NAME "DEMO1" AS "DEMO2"

実行結果……"DEMO1" のプログラムの名前が "DEMO2" となる

説　明 これから変更しようとするファイルは既に存在するものでなければなりません。また、つけようとする名前と同じファイルが既に存在しているとエラーとなります。

〈現在のファイル名〉には、ドライブ番号（A～）を指定できます。

NAMEコマンドが実行されると、そのファイルの名前は変更されますが、フロッピーディスク上で、ファイルの存在する位置や、占有する大きさ等は変化しません。

NEW

new（ニュー：新しい）　　コマンド

働　き プログラムを削除し、すべての変数をクリアする。

書き方 NEW

説　明 NEWコマンドは、新しいプログラムを入力する前に実行します。

メモリにあるベーシックのプログラムを消し、すべての変数をクリア（初期状態にする）し、開かれているファイルがあれば、それも自動的にすべて閉じます。また、DEF FN文による定義も解消されます。

プログラム中でNEWコマンドを実行するとプログラムを消しダイレクトモードに戻ります。（ファイルは閉じません）。

参　照 DELETE、ERASE

サンプルプログラム

プログラムを消します。

```
10 ' ** NEW **
20 PRINT "PROGRAM ヲ ケシマス "
30 FOR E=1 TO 30:NEXT
40 NEW
```

OCT$

octal$ （オクト ドル：8進数の文字列） 関数

働 き 数値を8進数の文字列に変換します。

書き方 OCT$ （〈数式〉）

例 LPRINT OCT$(10) 実行結果……12と表示される

説 明 〈数式〉の値を8進数の文字列に変換し、その結果を得ます。〈数式〉の値の範囲は－32768から32767までです。関数のため、他の命令と組み合せて使用します。

変換結果の文字列はゼロサプレス（先頭の不用な0を取り除く）されています。

参 照 BIN$，HEX$

ON ERROR GOTO

on error goto（オンエラーゴーツ：エラーが発生したとき） ステートメント

働 き エラー処理ルーチンの開始行を定義する。

書き方 ON ERROR GOTO 〈行番号〉

例 ON ERROR GOTO1000

説 明 ON ERROR GOTO文を実行しておくと、エラーが発生したときに割り込みがかかり、指定した行番号から始まるプログラム（エラー処理ルーチン）へジャンプします。

通常、エラーが発生すると、BASICで用意しているエラーメッセージが表示され、プログラムの実行を中断します。しかし、ON ERROR GOTO文を実行してあると、エラーが発生したときにエラーメッセージが表示されず、〈行番号〉で指定したエラー処理ルーチンへジャンプします。このときもし〈行番号〉の行がプログラム上に存在しなければ“Undefined line”エラーが起こります。

〈行番号〉で始まるエラー処理ルーチンでは任意のエラー処理を行い、RESUME文でエラー回復処理をして終了します。このエラー処理ルーチンではERR関数やERL関数を使ってエラーコードやエラーの発生した行番号を知ることができます。

ON ERROR GOTO 0を実行すると、ON ERROR GOTO〈行番号〉での定義を無効にし、以降、エラーが発生すれば、用意しているエラー処理が行われます。

ON ERROR GOTO文は、別のON ERROR GOTO文やRUN文、CLEAR文を実行するまで有効です。

注意！ ON ERROR GOTO文は、別のON ERROR GOTO文やRUN文、CLEAR文、NEW文を実行するまで有効です。このため一度処理ルーチンを設定してダイレクトモードでエラーが発生すると、プログラムが実行されてしまいます。プログラムの最後には、ON ERROR GOTO 0を実行してください。

エラー処理サブルーチンを実行中にエラーが起きた場合には、それに対するエラーメッセージが表示され、実行が停止されます。エラー処理サブルーチンの中ではエラー割り込みは起こりません。

参 照 RESUME、ERROR、ERR/ERL、第1章「割込み」

ON GOTO/ON GOSUB

on goto　（オン ゴーツー：行番号へジャンプ）
on gosub　（オン ゴーサブ：サブルーチンへジャンプ）　ステートメント

働き 〈式〉の値により、指定された行番号、サブルーチンへジャンプします。

書き方 ON〈式〉GOTO　〈行番号〉〔,〈行番号〉・・・〕
ON〈式〉GOSUB〈行番号〉〔,〈行番号〉・・・〕

例 ON A GOTO 100, 200, 300, 400
ON A GOSUB 100, 200, 300, 400

説明 〈式〉の値は、GOTO文またはGOSUB文に続く〈行番号〉の何番目にジャンプするのかを示しています。たとえば、〈式〉の値が3であったなら、左から3番目の〈行番号〉へジャンプします。

〈式〉の値が負になった場合には、"Illegal function call"エラーが起こりますが、値が0または〈行番号〉の個数より大きくなった場合には、ON GOSUB/ON GOTO文の次の行を実行します。

参照 GOSUB、GOTO

サンプルプログラム

入力した数値の大きさによって分岐します。

```
10 ' ** ON GOTO **
20 INPUT A
30 B=INT(A/10)
40 IF B>3 GOTO 100
50 ON B GOTO 70,80,90
60 PRINT "10 ﾖﾘ ﾁｲｻｲ":END
70 PRINT "10 ｶﾗ 19":END
80 PRINT "20 ｶﾗ 29":END
90 PRINT "30 ｶﾗ 39":END
100 PRINT "40 ﾖﾘ ｵｵｷｲ"
110 END
```

ON INTERVAL GOSUB

on interval gosub〔オン インターバル ゴーサブ:
タイマ割込みでジャンプ〕　ステートメント

働　き タイマ割込み時間を設定し、割込みのジャンプ先を定義します。

書き方 ON INTERVAL=〈時間〉 GOSUB 〈行番号〉

例 ON INTERVAL=60 GOSUB 200

説　明 〈時間〉に設定された数の1/60秒ごとに〈行番号〉で始まるプログラム(割込み処理ルーチン)へジャンプします。

〈時間〉は1/60秒単位で、1～65535までの数値です。
〈行番号〉は割込み処理ルーチンの開始行番号です。

割り込みを有効にするには、INTERVAL ON命令を実行しておく必要があります。なお、割込み処理ルーチンを実行している間は、自動的にINTERVAL STOP状態になり、RETURN文実行とともにINTERVAL ON状態に戻ります。

割込み処理ルーチンからの復帰は、GOSUB文で呼び出すサブルーチンと同じようにRETURN文を使います。

割込みはプログラム実行中でのみ有効です。また、ON ERROR文で定義するエラー割込み処理中、あるいは他の割込み処理中(STRIG, STOP, SPRITE, KEY)では、これらの処理ルーチンが終了するまで割込みは保留されます。

参　照 INTERVAL ON/OFF/STOP、第1章「割込み」、第2章

O

サンプルプログラム　1秒ごとのタイマー

```
10 ' ** ON INTERVAL **
20 ON INTERVAL=60 GOSUB 50
30 CLS:TIME=0:INTERVAL ON
40 GOTO 40
50 M=(TIME/3600) MOD 60
55 S=(TIME/60) MOD 60
56 D=TIME MOD 60
60 LOCATE 5,5:PRINT USING"##分##秒";M,S
70 RETURN
```

ON KEY GOSUB

on key gosub (オン キー ゴーサブ：ファンクションキーで割込み)　ステートメント

働き ファンクションキーによる割込みのジャンプ先を定義します。

書き方 ON KEY GOSUB〔〈行番号〉〕〔,〈行番号〉・・・〕

例 ON KEY GOSUB 100, 200, 300, 400

説明 この命令は、ファンクションキーによる割込みが発生したときに、どの割込み処理ルーチンにジャンプするかを定義します。割込みを有効にするには、KEY(n)ON命令を実行しておく必要があります。

〈行番号〉は、割込みがかかったときにジャンプする処理ルーチンの開始行番号で、この並びの順序は、ファンクションキーのキー番号と1対1に対応しています。

割込み処理ルーチンからの復帰はGOSUB文で呼び出すサブルーチンの場合と同じようにRETURN文を使います。

注意！ 割込みはプログラム実行中でのみ有効です。またON ERROR文などと同時に使用することはできません。

参照 KEY ON/OFF/STOP、第1章「割込み」

サンプルプログラム

```
10 '** ON KEY GOSUB **
20 FOR K=1 TO 10:KEY K,"":NEXT
30 ON KEY GOSUB 70,80,90,100,110
40 FOR K=1 TO 5:KEY (K) ON:NEXT
50 GOTO 50
60 '
70 PRINT "KEY 1 ON":RETURN
80 PRINT "KEY 2 ON":RETURN
90 PRINT "KEY 3 ON":RETURN
100 PRINT "KEY 4 ON":RETURN
110 PRINT "KEY 5 ON":RETURN
```

ON SPRITE GOSUB

on sprite gosub (オン スプライト ゴーサブ: スプライトで割込み)　ステートメント

働　き　**スプライトが重なったときに発生する割込みのジャンプ先を定義します。**

書き方　ON SPRITE GOSUB 〈行番号〉

例　ON SPRITE GOSUB 500

説　明　PUT SPRITE で画面に描いたスプライトが他のスプライトと重なったときに割込みが発生し、〈行番号〉で始まるプログラム（割込み処理ルーチン）へジャンプします。なお、割込みを有効にするには、SPRITE ON命令を実行する必要があります。

割込み処理ルーチンからの復帰は、GOSUB文で呼び出すサブルーチンと同じようにRETURN文を使います。

割込み処理ルーチンを実行している間は、自動的にSPRITE STOP状態になりスプライトの割込みは受けつけられず、RETURN文実行とともにSPRITE ON状態になります。

この命令はプログラム実行中でのみ有効です。また、ON ERROR GOTO文および ON KEY GOSUB, ON STOP GOSUB文などの割込み処理を実行中は、これらの処理ルーチンが終了するまで割込みは保留されます。

MSX2 のSCREEN 4～8 のスプライトで、特殊なカラーコードを指定したものは衝突を起こしません。くわしくは、COLOR SPRITEを参照してください。（SCREEN 7 と8は、VRAMが128KBの機種でのみ有効です。）

参　照　SPRITE ON/OFF/STOP，PUT SPRITE，COLOR SPRITE，第2章

O

ON STOP GOSUB

on stop gosub (オン ストップ ゴーサブ：ストップキーの割込み)　**ステートメント**

働　き CTRL + STOP キーによる割込みのジャンプ先を定義します。

書き方 ON STOP GOSUB 〈行番号〉

例 ON STOP GOSUB 1000

説　明 この命令を実行すると、以後、STOP ON状態にしておけば、CTRL + STOP キーが押されたときに割込みが発生し、〈行番号〉で始まるプログラム（割込み処理ルーチン）へジャンプします。

割込み処理ルーチンからの復帰はGOSUB文で呼び出すサブルーチンと同じようにRETURN文を使います。

割込み処理ルーチンを実行している間は自動的に STOP STOP 状態になり，CTRL + STOP キーの割込みは受けつけられず、RETURN文実行とともにSTOP ON状態になります。

この命令はプログラム実行中でのみ有効です。また、ON ERROR GOTO文、ON KEY GOSUB文で定義する割込み処理を実行中には、これらの処理ルーチンが終了するまで割込みは保留されます。

注意！ この命令はプログラムの実行中に、誤ってストップキーを押してしまい、プログラムの実行が中断されてしまうのを防止するためのものです。したがって不用意にこの命令を実行すると、プログラムの実行を停止する CTRL + STOP キーの機能が失われ、システムリセットを行わない限りプログラムを中断することができなくなってしまいます。

参　照 ON KEY GOSUB，STOP ON/OFF/STOP，第１章「割込み」

サンプルプログラム

電源を切る以外に止まらないプログラム

```
10 '** ON STOP GOSUB **
20 PRINT "ﾃﾞﾝｹﾞﾝ ｦ ｷｯﾃｸﾀﾞｻｲ"
30 ON STOP GOSUB 60
40 STOP ON
50 GOTO 50
60 RETURN
```

ON STRIG GOSUB

on strig gosub(オン スティックトリガ ゴーサブ:トリガボタンの割込み)ステートメント

働　き ジョイスティックのトリガボタンが押されたときに発生する割込みの、ジャンプ先を定義します。

書き方 ON STRIG GOSUB〔〈行番号〉〕〔,〈行番号〉・・・〕

例 ON STRIG GOSUB 100, 200

説　明 この命令を実行すると、以後、STRIG ON状態にしておけばスペースキーあるいはジョイスティックのトリガボタンが押されたときに割込みが発生し〈行番号〉で始まるプログラム(割込み処理ルーチン)へジャンプします。

〈行番号〉は最大5個まで指定することができ、左側から順番に次のトリガボタンと対応しています。

0:キーボードのスペースキー
1:ジョイスティック1のトリガボタン1　　2:ジョイスティック2のトリガボタン1
3:ジョイスティック1のトリガボタン2　　4:ジョイスティック2のトリガボタン2

ただし、ジョイスティックによってはトリガボタン2がないものもありますので、1と3, 2と4の指定が同じになることもあります。

割込み処理からの復帰は、GOSUB文で呼び出すサブルーチンと同じようにRETURN文を使います。

割込み処理ルーチンを実行している間、自動的にSTRIG STOP状態になり、トリガボタンによる割込みは受けつけられず、RETURN文実行とともに、STRIG ON状態になります。

注意! この命令はプログラム実行中でのみ有効です。また、ON ERROR GOTO文、ON KEY GOSUB文、ON STOP GOSUB文、ON SPRITE GOSUB文で定義する割込み処理を実行中はこれらの処理ルーチンが終了するまで割込みは保留されます。

参　照 STRIG ON/OFF/STOP, 第1章「割込み」、第2章

サンプルプログラム

“BOMB!!” の表示とともに発射音を出します

```
10 '** ON STRIG GOSUB **
20 ON STRIG GOSUB 50
30 STRIG(0) ON
40 GOTO 40
50 SOUND 0,0 :SOUND 1,0
60 SOUND 6,1 :SOUND 7,7
70 SOUND 8,16:SOUND 11,100
80 SOUND 12,100:SOUND 13,0
90 PRINT "BOMB!!":RETURN
```

O

OPEN

open （オープン：開く）　　ステートメント/Disk BASIC

働き　プログラムで使用するために、入出力用のファイルを開きます。

書き方　OPEN "〈ファイルスペック〉"〔FOR〈モード〉〕AS#〈ファイル番号〉

例　OPEN "CAS：MSX" FOR OUTPUT AS#1

説明　〈ファイルスペック〉で指定したファイルを開き、INPUT#文やPRINT#文でデータを入出力できるようにします。オープンするファイルには〈ファイル番号〉を割り当て、ファイルへの入出力は〈ファイル番号〉を指定して行ないます。

〈ファイルスペック〉の形式は "〈デバイス名〉：〈ファイル名〉" です。

〈デバイス名〉は次のような種類があります。

デバイス名	入出力デバイス	使用できるモード		
		INPUT	OUTPUT	APPEND
CAS：	カセットレコーダ	○	○	×
CRT：	テキスト画面	×	○	×
GRP：	グラフィック画面	×	○	×
LPT：	プリンタ	×	○	×
	フロッピーディスク	○	○	○
MEM：	RAMディスク（MSX2のみ）	○	○	○

モードには次の3種類があります。

INPUT　既存のファイルから入力を行います。
OUTPUT　新しくファイルを作り、出力を行います。
APPEND　既存のファイルに、出力を行います。

〈ファイル名〉は1～6文字の文字列で表しますが、デバイス名がCRT，GRP，LPTの場合には指定する必要はありません。

〈ファイル番号〉は、1～15の整数を指定できますが、MAX FILES文で指定した数を越えて指定することはできません。また、既にオープンしているファイル番号を指定することはできません。INPUT#やPRINT#命令の入出力には、ここで指定したファイル番号を使います。

OPEN文は、指定したファイルを入出力する際に使うバッファ領域を確保し、ファイルをCLOSE文で閉じるときに解放します。〈ファイル番号〉は、このバッファ領域と対応しています。

RAMディスクは、MSX2のみ使用でき、RAMディスクの使用前にCALL MEMINI命令を実行します（CALL MEMINI参照）。

注意！　FOR〈モード〉を省略したとき

フロッピーディスクを接続していないときはエラーになりますが、フロッピーディスクを使用しているときはランダムファイルに対して入出力を行うことを意味します。

シーケンシャルファイルへのINPUTモード及びAPPENDモードでは、指定されたファイルが存在しないと、"File not found" エラーとなります。
シーケンシャルファイルのOUT PUTモードでは、常に指定された名前のファイルを新しく作り、同一名のファイルがあった場合にはそのファイルは削除されます。ランダムファイルのOUT PUTモードでファイルが存在しない場合には新たに作られますが、INPUTモードでは "File not found" エラーとなります。

参　照　CLOSE, INPUT#, PRINT#, MAXFILES, INPUT$, EOF

サンプルプログラム

画面に適当な文字を表示します。

```
10 ' ** OPEN  **
20 COLOR ,1,1:SCREEN 3
30 OPEN "GRP:" FOR OUTPUT AS #1
40 FOR S=1 TO 20
50 X=RND(1)*200+30
60 Y=RND(1)*150+30
70 C=RND(1)*15
80 R$=CHR$(RND(1)*223+32)
90 PSET(X,Y),1
100 COLOR C:PRINT #1,R$
110 NEXT :CLOSE
120 END
```

OUT

out（アウト：出力する） ステートメント

働き 出力ポートに1バイトのデータを送ります。

書き方 OUT <ポート番号>，<式>

例 OUT &HFF，&H80　　実行結果……&HFFの出力ポートへ&H80というデータを出力

説明 <ポート番号>は出力ポートの番号、そして<式>は出力する1バイトのデータです。どちらのパラメータも0～255(&H0～&HFF)の整数で指定します。(<ポート番号>が255を越えてもエラーにはなりません)
OUT文は、入出力ポートの使い方について知識を得てから使用してください。

参照 INP，I/Oマップ

PAD

pad（パッド：当て物、タッチパッド） 関数

働き タッチパッドなどの状態を調べます。

書き方 PAD（<式>）

例 X＝PAD(1)

説明 指定した<式>の値により、次のようなタッチパッドの状態を返します。PAD(0)またはPAD(4)が評価されたときにはじめて、そのタッチパッドの座標値が有効になりますので、2個のタッチパッドを同時に接続して使用した場合、ジョイスティック1に接続したタッチパッドの状態が、ジョイスティック2に接続したタッチパッドの座標値に悪影響を及ぼすことがあります。
そのため、2つのタッチパッドの値を連続して読むときには、2つのPAD関数の間に、何行かのプログラムを入れるか、待ちループを入れます。
<式>の8～19はMSX2の機種で、機器を使用できるもののみ有効です。

<式>の値	タッチパッドの接続されたジョイスティック番号	返される値の内容	返される値
0	1	タッチパッドに触れている 触れていない	−1 0
1	1	タッチパッドのX座標 (押されていないときは0)	座標値
2	1	タッチパッドのY座標 (押されていないときは0)	座標値
3	1	タッチパッドのスイッチが 押されている 押されていない	 −1 0
4	2	タッチパッドに触れている 触れていない	−1 0
5	2	タッチパッドのX座標 (押されていないときは0)	座標値
6	2	タッチパッドのY座標 (押されていないときは0)	座標値
7	2	タッチパッドのスイッチが 押されている 押されていない	 −1 0
8～11	—	ライトペンの情報	
12～15	1	マウスまたはトラックボールの情報	
16～19	2	マウスまたはトラックボールの情報	

PAINT

paint （ペイント：色を塗る） ステートメント

働　き　指定された境界色で囲まれた領域を塗りつぶします。

書き方　PAINT ｜(X, Y) / STEP (X, Y)｜〔,〈領域色〉〕〔,〈境界色〉〕

例　PAINT (50, 120), 3, 4

説　明　〈境界色〉で囲まれた領域の中、または外を指定された〈領域色〉で塗りつぶします。
(x, y) が領域の内側にあるときは中を、外側にあるときは、外を塗りつぶします。(x, y)の位置は、STEPを付けるとLP(最終参照点)からの相対座標となります。

〈領域色〉,〈境界色〉はともにカラーコードで指定します。
〈領域色〉を省略した場合には、COLOR文で指定した前景色が用いられます。〈境界色〉を省略した場合には〈領域色〉が〈境界色〉として用いられます。

〈境界色〉はSCREEN 3とMSX2のSCREEN 5～8でのみ使用することが可能で、SCREEN 2とSCREEN 4（MSX2）では無視され、〈領域色〉と同じ色が〈境界色〉とみなされます。

(X, Y)は塗り始める座標を指定します。もしこの点が、指定された境界色と既に同じ色であった場合には、PAINTは塗りつぶしを行いません。

PAINT文はグラフィックモードでのみ使うことができます。
SCREEN 4～8はMSX2でのみ有効です。またSCREEN 7, 8は、VRAMが128KBの機種でのみ使用できます。

参　照　SCREEN, COLOR

サンプルプログラム

図を塗りつぶします。(10行をSCREEN 3にすると違いがわかります。)

```
5 '** PAINT **
10 COLOR 15,4,7:SCREEN 2
20 LINE(50,50)-(200,150),15,B
30 CIRCLE(100,100),40,15
40 CIRCLE(150,100),40,8
50 PAINT(100,100),15
60 PAINT(150,100),8
70 GOTO 70
```

←CTRLキーを押しながらSTOPキーで止まります。

O P

PDL

paddle（パドル:へら、足）　　関数

働き　パドルの状態を得る。

書き方　PDL（〈パドルの番号〉）

例　PRINT PDL（2）　　実行結果……ポート2パドルの回転を表示します。

説明　〈パドルの番号〉はパドルがポート1につながっていれば1、3、5、7、9または11を指定し、パドルがポート2につながっていれば2、4、6、8、10または12を指定します。パドルの状態は回転の度合によっては0～255の範囲の数値で得られます。

参照　PAD，STRIG

PEEK

peek（ピーク:のぞく、見る）　　関数

働き　メモリの指定された番地の内容を読み出します。

書き方　PEEK（〈番地〉）

例　A＝PEEK（&HA000）

説明　〈番地〉で指定されたメモリの内容を読み出します。読み出すデータは1バイト分（0～255の値です。

〈番地〉は、－32768～65535（&H0～&HFFFF）の値を持つ数または式で指定します。〈番地〉が負数ならば65536を加えたものが実際の番地となります。

PEEK関数はPOKE文と対応している命令です。関数ですので、他の命令と組み合せて使用します。

参照　POKE，メモリマップ

サンプルプログラム　　メモリ番地、&HC000から&HC100の内容を表示します。

```
10 '** PEEK **
20 FOR AD=&HC000 TO &HC100 STEP 8
30 PRINT "&H"+HEX$(AD)+"  ";
40 FOR CC=0 TO 7:M$=HEX$(PEEK(AD+CC))
50 PRINT RIGHT$("00"+M$,2)+"  ";
60 NEXT CC:PRINT
70 NEXT AD
80 END
```

PLAY

play （プレイ:演奏する） ステートメント

働　き　音楽を演奏します。

書き方　PLAY 〈文字式1〉〔,〈文字式2〉〕〔,〈文字式3〉〕

例　PLAY "CDEFGABO5C"
PLAY A$, B$, C$

説　明　〈文字式１〉、〈文字式２〉、〈文字式３〉で指定された音楽を演奏します。〈文字式１〉、〈文字式２〉、〈文字式３〉はそれぞれボイスチャンネル１、２、３に対応しており、３重和音まで出すことができます。

〈文字式〉は次のミュージックマクロ命令によって作られ、文字定数、文字変数、あるいはそれらを組合せた〈式〉でもかまいません。

ミュージックマクロ命令は、大きく分けて音の高さ、音の長さ、音の速さ(テンポ)、音の大きさ、音色、ローカルマクロの６種類があります。

●音の高さ

A～G〔 # | + | − 〕　音の高さを音階名（ド、レ、ミ、ファ、ソ、ラ、シ）で表現します。C～Gまでがド～ソ、AとBがラとシにそれぞれ対応しています。またA～Gの後ろに#や+記号を付けると半音上がった音になり、逆に−記号を付けると半音下がった音になります。（黒鍵盤に相当します。）

例）

PLAY "CEG"

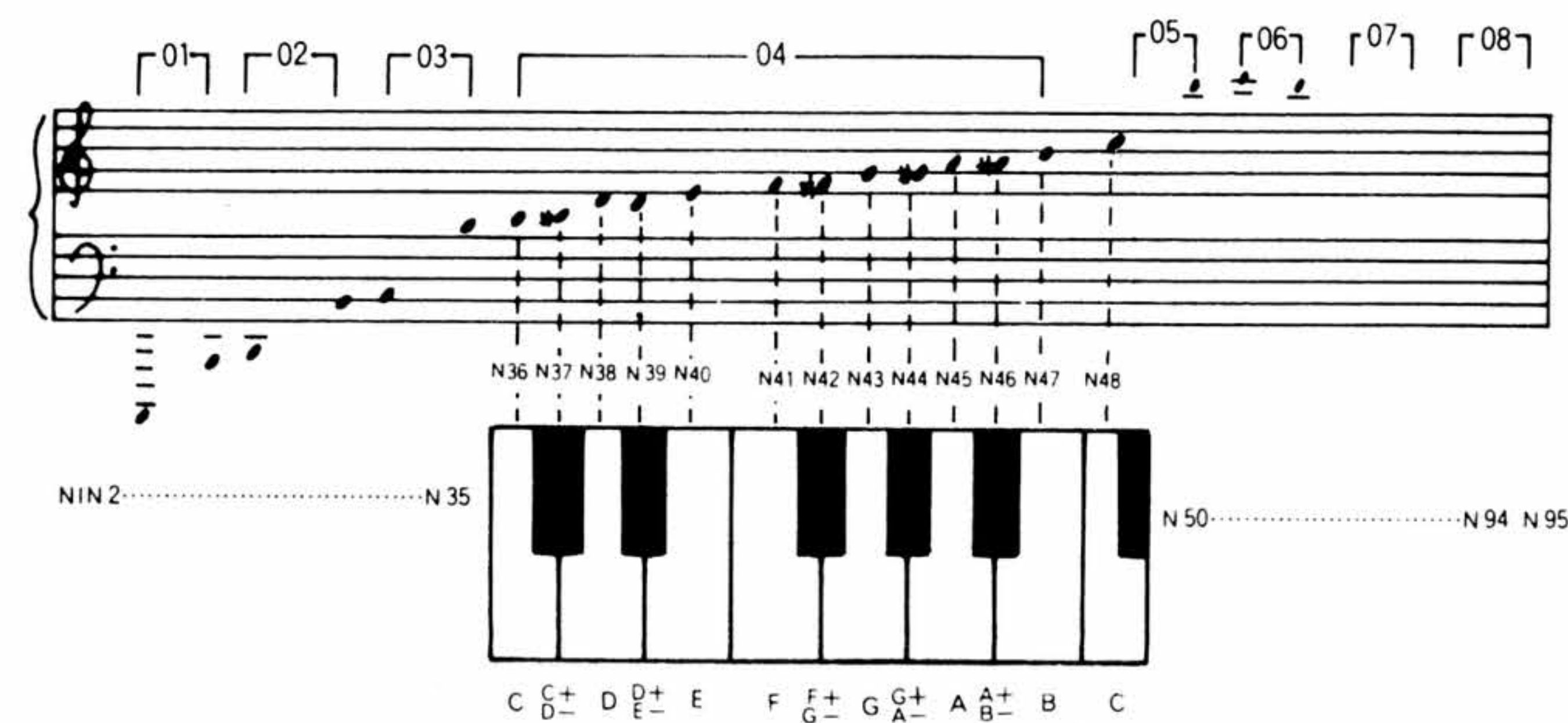
O〈数値〉　A～Gの音階のオクターブを１～８の数値で指定します。(８オクターブが指定できます。) 電源を入にしたときには、O4（ハ長調）に設定されており、１度オクターブの指定をすると、そのボイスチャンネルで次のオクターブ指定をするまで有効です。

PLAY

N〈数値〉

音の高さを0〜96の数値で指定します。N1はO1のC♯と同じ高さで、N2、N3……と半音ずつ上がって行き、N95はO8のBと同じになります。またN0はR(休符)を指定したことになります。

例)

PLAY "N36N37N38N39N40N41N42N43"

例 O 1

C	C+ D−	D	D+ E−	E	F	F+ G−	G	G+ A−	A	A+ B−	B
0	1	2	3	4	5	6	7	8	9	10	11

O 4

C	C+ D−	D	D+ E−	E	F	F+ G−	G	G+ A−	A	A+ B−	B
36	37	38	39	40	41	42	43	44	45	46	47

● 音の長さ

L〈数値〉

音の長さを $\frac{1}{〈数値〉}$ 音にします。〈数値〉の範囲は1〜64までです。

(全 音 符)=L1　　(2分音符)=L2
(4分音符)=L4　　(8分音符)=L8
(16分音符)=L16　　(32分音符)=L32
(64分音符)=L64

初期値はL4です。1度音の長さを指定しますと、次に音の長さを指定するまで有効です。また、1音についてだけ長さを変えたいときは、音階名の後ろに音の長さを表す〈数値〉を付け加えます。たとえばL16CはC16と同じです。

例)

PLAY "L4DFG8O5C8O4AG8O5CO4AFGDC"

・（ピリオド）　付点音符のように音階名や休符の後ろに付け加えて、本来の音の長さの1.5倍にします。また、2個並べて書きますと$\frac{9}{4}$倍、3個並べて書くと$\frac{27}{8}$倍となります。

例）

PLAY "S9M3000T200　L4F.FF.GA.AA.GF.FF.DC2"

R〔〈数値〉〕　休符です。〈数値〉はＬマクロ命令での指定方法と同じです。〈数値〉を省略するとR4とみなされます。休符の〈数値〉は、Ｌマクロ命令に影響されません。

●速さ（テンポ）

T〈数値〉　テンポを指定します。〈数値〉は1分間に演奏される4分音符の数です。〈数値〉は32〜255の範囲で指定します。初期値はT120です。1度速さの指定をすると、次に速さの指定をするまで有効です。

●音の大きさ

V〈数値〉　音量を指定します。〈数値〉の範囲は0〜15で、数が大きいほど音も大きくなります。初期値は8です。
なお、1度音量の指定をすると、次に音量の指定をするまで有効です。

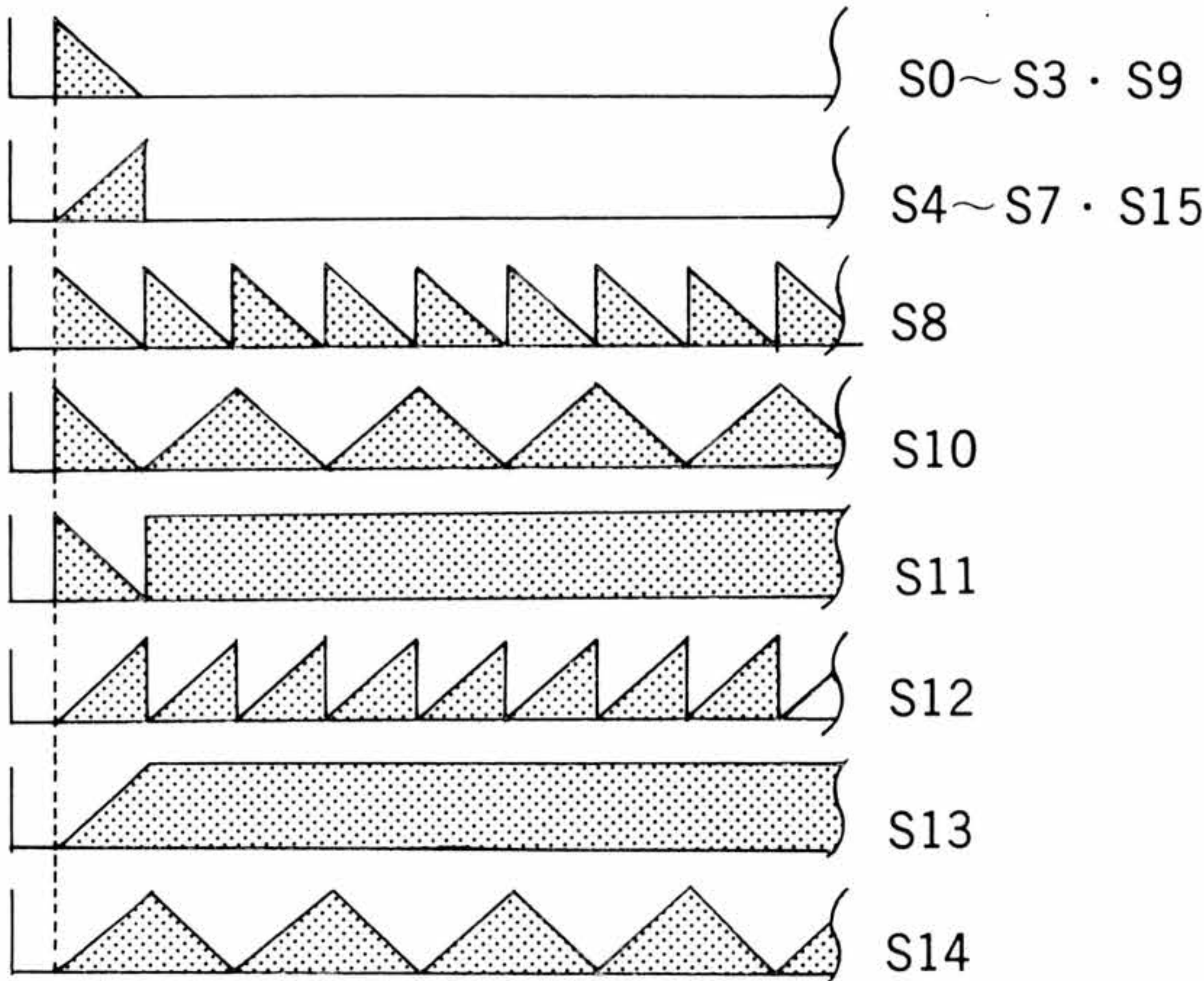
例）

PLAY "V15CGV0CG"

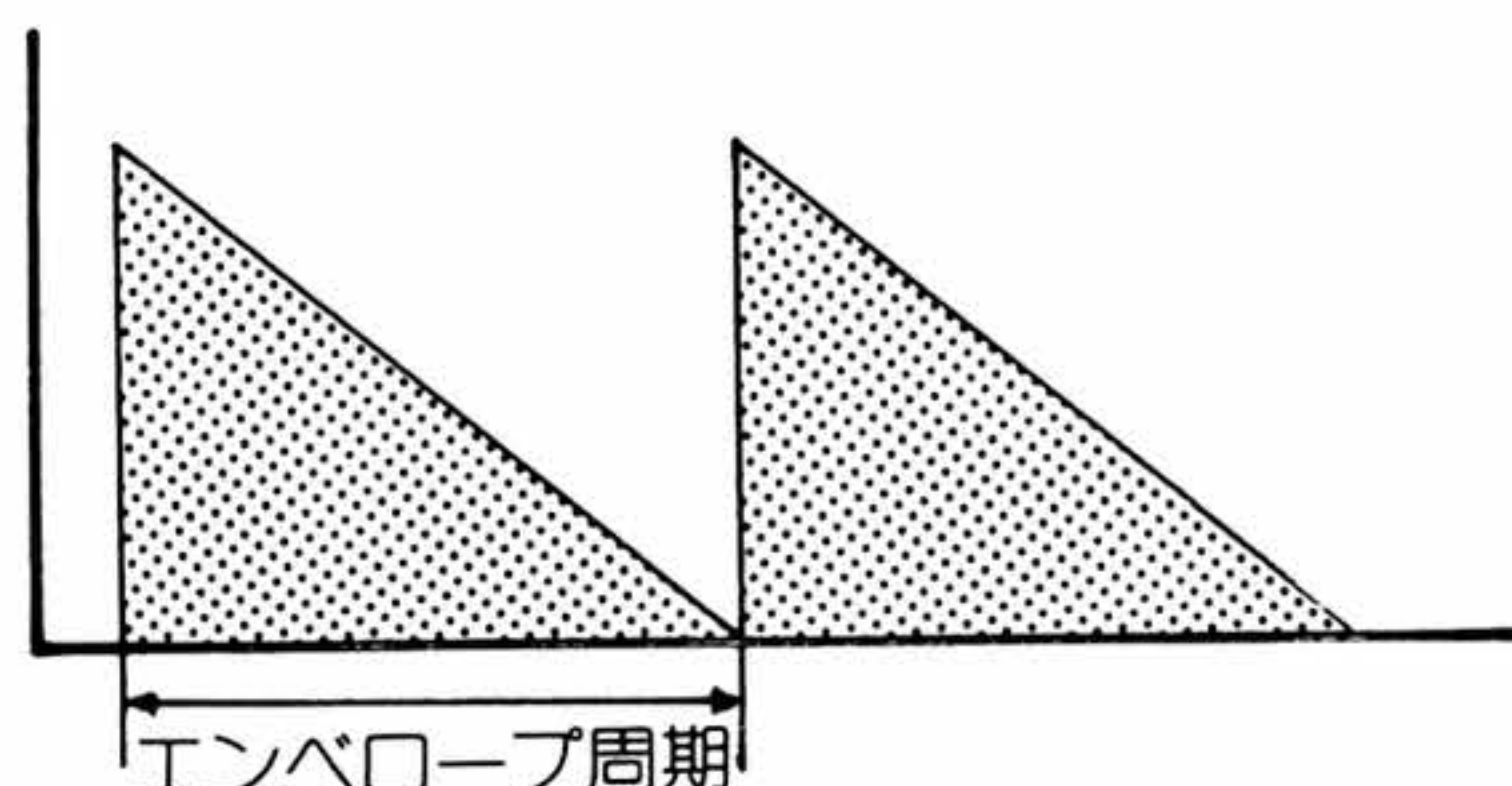
●音色

音色はエンベロープ周期（Mマクロ）とエンベロープ形状（Sマクロ）とでほぼ決まります。なおSマクロ命令は、Mマクロ命令とともに使用し、Vマクロ命令と同時に使うことはできません。

S〈数値〉　エンベロープ形状（音量変化の波形）を指定します。〈数値〉は1〜15が指定でき、波形は次の図のように対応しています。

S0〜S3・S9
S4〜S7・S15
S8
S10
S11
S12
S13
S14

PLAY

M〈数値〉　　エンベロープ周期を指定します。エンベロープ周期は0～65535までの〈数値〉で指定します。初期値は255です。

エンベロープ周期

● ローカルマクロ

X〈文字変数〉;　〈文字変数〉中の文字列に含まれているミュージックマクロ命令を実行します。〈文字変数〉には上記のミュージックマクロをあらかじめ入れておきます。〈文字変数〉の後のセミコロン（;）は必ず入れてください。

例）

```
A$= "CCGGAAG" :B$= "FFEEDDC"
PLAY "XA$; XB$;"
```

ミュージックマクロ命令では〈数値〉を定数で指定しますが、"=〈変数名〉;"の形式にすれば数値変数を指定することができます。
この場合〈変数名〉の後ろにはセミコロン（;）を必ず入れます。

例）

```
10 FOR S=1 TO 96
20 PLAY "N=S;"
30 NEXT S
```

参　照　第2章、第4章「サンプルプログラム」

PLAY

play（プレイ：演奏している）　関数

働き　音楽を演奏中かどうかを調べます。

書き方　PLAY（〈ボイスチャンネル番号〉）

例　A＝PLAY（0）

説明　各ボイスチャンネルがPLAY文で命じた音楽を演奏中かどうかを調べ、演奏中であれば－1、そうでなければ0を得ます。〈ボイスチャンネル番号〉は次のように指定します。

0：ボイスチャンネル1、2、3（いずれかが演奏中であれば－1を得ます）
1：ボイスチャンネル1
2：ボイスチャンネル2
3：ボイスチャンネル3

POINT

point（ポイント：点）　関数

働き　指定された座標の点の色を調べます。

書き方　POINT（X，Y）

例　C＝POINT（50，50）

説明　（X，Y）で指定した座標上に表示されている点の色を調べ、結果をカラーコード（0～15）で得ます。

POINT関数はグラフィックモードでのみ有効です。また、指定された座標上にスプライトパターンが表示されていても、スプライトパターンの色を調べることはできません。

サンプルプログラム

```
10 '** POINT **
20 COLOR ,1,1:SCREEN 2
30 FOR P=1 TO 100
40 C=RND(1)*13+2
50 X=RND(1)*255:Y=RND(1)*191
60 LINE(X,Y)-(X+3,Y+3),C,BF
70 NEXT P
80 OPEN "GRP:" FOR OUTPUT AS#1
90 FOR S=1 TO 50
100 X=RND(1)*255:Y=RND(1)*191
110 C=POINT(X,Y)
120 PSET(X,Y),1:PRINT #1,C
130 NEXT
140 GOTO 140
```

POKE

poke（ポーク：突く）　ステートメント

働　き 指定した番地のメモリにデータを書き込みます。

書き方 POKE 〈番地〉，〈データ〉

例 POKE &HF000，&HFF

説　明 指定されたメモリの番地に1バイト（8ビット）のデータを書き込みます。〈番地〉は－32768～65535（&H0～&HFFFF）の値を持つ式で指定します。また、〈番地〉が負数ならば65536を加えたものが実際の番地となります。

〈データ〉はメモリに書き込まれるデータで、0～255（&H0～&HFF）の値を持つ数式です。もし小数点以下があれば整数化してから実行されます。

この命令は、現在のメモリの内容を書き換えてしまうため、不用意に使うとBASICが使っている作業領域（&HF380以降）を壊してしまい、誤動作の原因となることがあります。使う時にはメモリマップなどで使用可能な領域かどうかを確認してください。

参　照 CLEAR，PEEK，第5章「メモリマップ」

POS

position（ポジション：位置）　システム変数

働　き テキスト画面上のカーソルの水平位置の値を得ます。

書き方 POS（〈式〉）

例 P=POS（0）

説　明 〈式〉はダミーの引数で、何を指定しても同じですが、通常0を使います。得られる値は0～31（SCREEN 0のとき0～39）です。画面の左端が0です。
この関数はテキストモードでのみ有効です。

参　照 CSRLIN

サンプルプログラム

```
10 CLS
20 LOCATE 5,5
30 PRINT "カーソルの いち";
40 X=POS(0)
50 Y=CSRLIN
60 PRINT X;Y
```

```
カーソルの いち 13  5
Ok
```

PRESET

point reset （ポイントリセット：点の消去） ステートメント

働　き 指定された座標に描かれた点を消します。

書き方 PRESET ｜(X, Y) / STEP (X, Y)｜〔,〈カラーコード〉〕

例 PRESET (100, 100)

説　明 (X,Y)で表わされる位置に描かれた点を消します。STEPを付けた場合はLP(最終参照点)からの相対座標となります。PRESET文を実行した結果、LPは指定した座標へ移動します。

〈カラーコード〉は点を消すときに用いる色を指定します。省略した場合にはCOLOR文で指定された背景色が採用されます。

参　照 PSET, COLOR

サンプルプログラム

画面を一度、青色で塗り、黒色にもどす

```
10 '** PRESET **
20 COLOR 15,1,7:SCREEN 3
30 LINE(0,0)-(255,191),4,BF
40 X=RND(1)*255
50 Y=RND(1)*191
60 PRESET(X,Y)
70 GOTO 40
```

P

PRINT

print（プリント：表示する）　ステートメント

働き テキスト画面に文字や記号を表示します。

書き方

PRINT〔〈式〉〔 ; , 〈式〉・・・・〕〕

例

PRINT "ABC"; "DEF"　　実行結果……ABCDEFと表示される。
? 7*6　　実行結果……42

説明 〈式〉で指定する数値や文字列を画面に表示します。〈式〉が省略されている場合は改行のみを行います。

複数個の〈式〉を指定する場合は式の間をカンマ(,)またはセミコロン(;)で区切ります。

区切り記号にセミコロンを使うと直前にプリントしたもののすぐ後ろに続いて次の値や文字列が表示されます。また、一番最後の〈式〉の後ろにセミコロンがあれば、次に実行されるPRINT文で出力する〈式〉の内容がすぐ後に表示されます。

式の値や文字列を表示するスペースは、あらかじめ各行が14文字毎に分割して定められており、区切り記号にカンマを使うと次の領域の始めから表示され、頭ぞろえを行うことができます。(TAB命令に類似しています)。

数値を表示した場合、その後ろに空白が1つつけ加えられ、数値の前には符号を表示するための桁が確保され、数値が負の値はマイナス符号（正の場合は空白）が表示されます。

表示しようとする数値が現在のカーソルの位置より後方に表示しきれない場合（それを表示すると次の行に渡ってしまうとき）改行してそれを表示します。

この"PRINT"の簡略形として疑問符（?）を使用することができます。プログラムの入力時に疑問符(?)を使用しても、リストアウトすると"PRINT"に置き換えられます。

PRINT文はテキスト画面でのみ有効です。

サンプルプログラム

すべてのキャラクターを表示します

```
10 '** PRINT **
20 FOR C=&H20 TO &HFF
30 PRINT CHR$(C);" ";
40 NEXT :PRINT
50 FOR C=&H40 TO &H5F
60 PRINT CHR$(1)+CHR$(C);" ";
70 NEXT
```

PRINT USING

print using (プリントユージング)　ステートメント

働き 文字列や数値を指定した〈書式〉で表示します。

書き方 PRINT USING 〈書式〉；〈式〉・・・・

例 PRINT USING "###.##";A

説明 〈書式〉に用いられる書式制御文字に従って、〈式〉で表す文字列や数値をテキスト画面に表示します。

〈式〉はカンマ(,)やセミコロン(;)で区切って複数個指定することができます。この場合、左側の〈式〉から順番に画面へ表示されます。

●文字列を編集する書式制御文字

！…………与えられた文字変数や文字列の左側1字を表示します。

例）

```
10A$= "MSX"
20PRINT USING "!";A$
run
Ok
```

&〈n個の空白〉&

文字列の左側から（n＋2個の）文字を表示します。文字列が（n＋2）文字より長ければ余分な文字は表示されず、短ければ足りない文字数だけの空白を右側に補って表示されます。

例）

```
A$= "Japan"
Ok
PRINT USING "&  &";A$
Jap
Ok
```

@…………〈書式〉の文字列中に含まれる"@"を〈式〉の文字列で置き換え、〈書式〉を画面に表示します。1つの"@"は左側の〈式〉から順番に対応しており、〈書式〉中の"@"の数が〈式〉の数より多い場合、残りの"@"は無視されます。

例）

```
10 A$= "I":B$= "CAT"
20 PRINT USING "@  Like @.";A$,B$
I Like CAT
Ok
```

PRINT USING

●数値を編集する書式制御文字

#と.…… 数値を"#"で指定した桁数だけ表示します。数値が指定した桁数より小さければ右づめで表示され、左側に空白が補われます。
また、ピリオド(.)は小数点の位置を指定します。小数部の桁数が書式で指定した桁数より小さければ、右側に0が補われます。

例)

```
PRINT USING "###.##"; 10.2, 2, 3.345,.24
 10.20  2.00   3.35   0.24
Ok
```

+と−……〈書式〉の左端に"+"を付けると、数値の前に正負符号が表示され、〈書式〉の右端に"+"を付けると数値の後ろに正負符号が表示されます。また、〈書式〉の右端に"−"を付けると、数値が負の場合に数値の後ろに"−"が表示されます。〈書式〉の左端に付けたり、2個以上並べたときには制御文字とみなされません。

例)

```
PRINT USING "+###.##";1.25,-1.25
 +1.25   -1.25
Ok
PRINT USING "###.##+";1.25,-1.25
 1.25+   1.25-
Ok
```

＊＊………〈書式〉の左端を"##"の代わりに"＊＊"にしておくと、数値の整数部の桁数が書式で指定した桁数より小さければ、左側に"＊"が補われます。

例)

```
PRINT USING "**##.##";1.25, -1.25
***1.25**-1.25
Ok
```

¥¥………〈書式〉の左端に"¥¥"を付けておくと、数値の直前に"¥"が表示されます。"¥¥"は"##"を指定したのと同じように2桁分の表示桁を確保しますが、符号(+、−)を表示するために、このうちの1桁が使われます。"¥¥"は指数形式の書式指定をしているときには使用できません。

例)

```
PRINT USING "¥¥###.##";12.35, -12.35
 ¥12.35   -¥12.35
Ok
PRINT USING "¥¥###.##-";12.35, -12.35
 ¥12.35   ¥12.35-
Ok
```

＊＊¥…… "＊＊"、"¥¥"と同じように使います。"＊＊¥"は"＊＊＊"を指定したのと同じように3桁分の表示桁を確保しますが、"¥"を表示するために、このうちの1桁が使われます。"＊＊¥"は指数形式の書式指定をしているときには使用できません。

例)

```
PRINT USING "＊＊¥#.##";12.35
＊¥12.35
Ok
```

,…………","を〈書式〉の整数部の"#"の並び（ピリオドの左側）におくと、数値の整数部が3桁ごとに","で区切られて表示されます。
","は指数形式の書式指定をしているときには使用できません。

例)

```
PRINT USING "####,.##";1234.5
1,234.50
Ok
```

∧∧∧∧…"∧∧∧∧"を桁数指定の"#"の並び（仮数部の数）の後に付けると、数値が指数形式で表示されます。また、〈書式〉の先頭に"+"、または〈書式〉の後に"+"か"−"を指定すると符号（正ならば空白、負ならば"−"）が表示されます。

例)

```
PRINT USING "##.##∧∧∧∧";234.56
 2.35E+02
Ok
```

PRINT USING

注意！●表示しようとする数値が〈書式〉で指定した桁数より大きい場合、数値の直前に“%”が表示されます。また、表示しようとする数値を丸めたことによって書式で指定した桁数より大きくなった場合も、数値の直前に“%”が表示されます。

例）

```
PRINT USING ".##";.999
%1.00
Ok
```

```
PRINT USING "##.##";123.45
%123.45
Ok
```

●〈書式〉の中に上記の書式制御文字以外の文字を置くと、文字の位置に応じて数値の前や後ろにその文字が表示されます。

例）

```
PRINT USING "TAX **¥###";123
TAX **¥123
Ok
```

PRINT＃

print＃(プリントシャープ:データの出力)　ステートメント/Disk BASIC

働　き ファイルにデータを書き出す。

書き方 PRINT　#〈ファイル番号〉〔,〈式〉〔 ; / , 〈式〉・・・・〕〕

例 PRINT　#2, A, B, C

説　明 〈ファイル番号〉で指定したファイルに〈式〉のデータを出力します。ファイルは、あらかじめOPEN文でOUTPUTモードを指定して開いておかなければなりません。
PRINT#文は書き出すファイルが“CRT:”（テキスト画面）や“GRP:”（グラフィック画面）であればPRINT文と同じ働きをし、“LPT:”（プリンタ）であればLPRINT文と同じ働きをします。また、“CAS:”（カセットテープ）やフロッピーディスクおよび“MEM:”・(RAMディスク）へ書き出したデータは、INPUT#文、LINE INPUT#文、INPUT＄関数を使って読み込むことができます。(RAMディスクは MSX2のみです。)

〈ファイル番号〉はOPEN文で指定した番号です。

〈式〉はファイルに書き出す数値、または文字列です。複数の〈式〉を書き出す場合は〈式〉と〈式〉の間をセミコロン（;）かカンマ（,）で区切ります。
〈式〉が数値であれば、PRINT#文を実行すると、文字列に変換された数値がPRINT文と同様に空白で区切って書き出され、最後に改行コード〔キャラクタコードCHR$(13)+CHR$(10)〕が書き出されます。

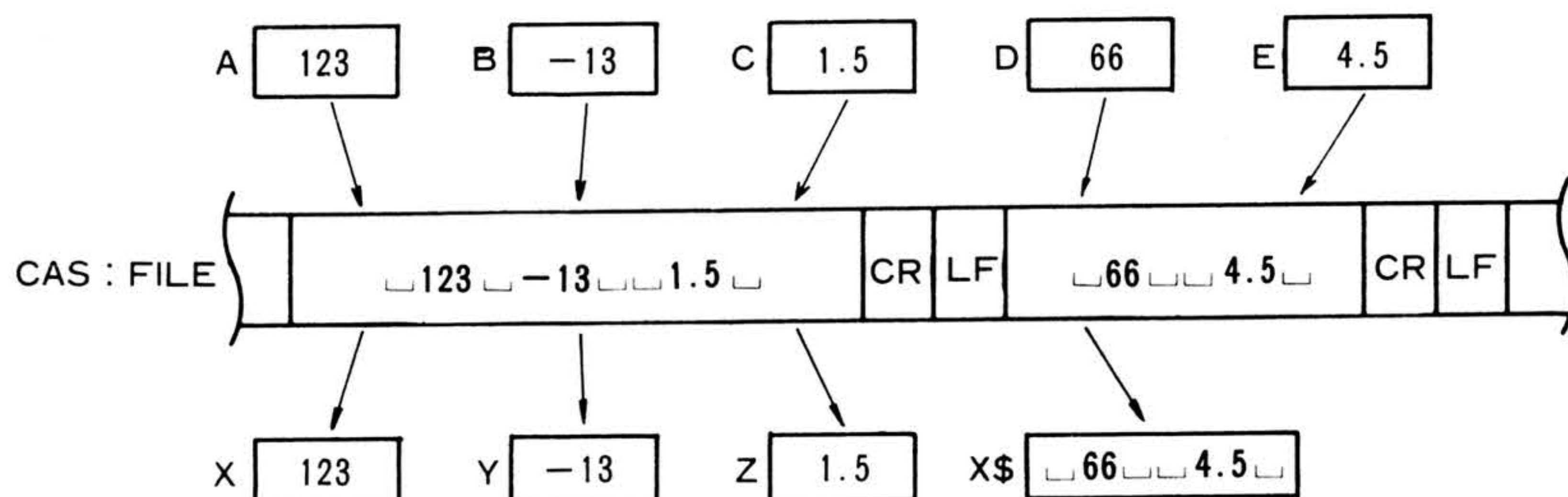
サンプルプログラム　カセットテープに「FILE」という名でデータを書きます。

```
10 OPEN "CAS:FILE" FOR OUTPUT AS#1
20 A=123:B=-13:C=1.5:D=66:E=4.5
30 PRINT #1,A;B;C
40 PRINT #1,D;E
50 CLOSE #1
60 END 'CONT ｼﾃｸﾀﾞｻｲ                  ←テープを巻きもどしてください。
70 OPEN "CAS:FILE" FOR INPUT AS#1
80 INPUT #1,X,Y,Z
90 LINE INPUT #1,X$
100 CLOSE #1
```

A 123　B −13　C 1.5　D 66　E 4.5

CAS : FILE　␣123␣−13␣␣1.5␣　CR　LF　␣66␣␣4.5␣　CR　LF

X 123　Y −13　Z 1.5　X$ ␣66␣␣4.5␣

P

PRINT # USING

print#using(プリントシャープユージング)　ステートメント/Disk BASIC

働き　文字列や数値を指定した書式でファイルに出力します。

書き方　PRINT # 〈ファイル番号〉, USING〈書式〉;〈式〉〔{ ; | , }〈式〉・・・・〕〕

例　PRINT #1, USING "@####. ##"; A$; B

説明　〈式〉で表す文字列や数値を〈書式〉に従って〈ファイル番号〉で指定したファイルへ出力します。PRINT# USING文は文字列や数値を〈書式〉に従って出力することを除けば、その機能はPRINT#と同じです。また、〈書式〉を使った出力方法はPRINT USING文と同じですので、そちらを参照してください。

参照　PRINT#, PRINT USING

PSET

point set (ポイントセット：点を打つ)　ステートメント

働き　指定した座標に点を描きます。

書き方　PSET { (X, Y) | STEP (X, Y) } 〔,〈カラーコード〉〕〔,〈論理演算子〉〕 (MSX2のみ)

例　PSET (100, 100), 8

説明　(X, Y)の位置に〈カラーコード〉で指定した色で点を描きます。STEPを付けるとLP(最終参照点)からの相対座標による指定となります。

〈カラーコード〉を省略した場合は、COLOR文の前景色が採用されます。また、MSX2では〈論理演算子〉を指定できます。これについてはLINEを参照してください。

PSET実行の結果、LPは指定された座標(X, Y)に移動します。
PSETステートメントはグラフィックモードでのみ使用できます。

参照　COLOR, PRESET, LINE

サンプルプログラム　適当に点を描きます。(止めるときは、CTRLキーを押してからSTOPキーを押す)

```
10 '** PSET **
20 COLOR ,1,1:SCREEN 3
30 C=RND(1)*13+2
40 X=RND(1)*255
50 Y=RND(1)*191
60 PSET(X,Y),C
70 GOTO 30
```

PUT

put（プット：出力する）　Disk BASIC

働　き　ランダム入出力用バッファ中のデータをランダムファイルへ出力します。

書き方　PUT〔#〕〈ファイル番号〉〔,〈レコード番号〉〕

説　明　〈ファイル番号〉で指定されたファイルに対応するバッファの内容を書き出します。指定されたファイルは、OPEN文でOUT PUTモードを設定されていなければなりません。

〈レコード番号〉は、1から4294967295までが有効で、バッファ中のデータが指定されたレコードに書き込まれます。〈レコード番号〉が省略された場合には、直前のGET文、PUT文で参照されたレコードの次のレコードに書き込まれます。

なお、書き出すデータはあらかじめFIELD文、LSET文/RSET文により準備しておかねばなりません。

参　照　FIELD, LSET/RSET

PUT KANJI

put kanji（プット・カンジ：漢字を出力する）　MSX2

働　き　SCREEN 5～8に漢字を表示します。

書き方　PUT KANJI〔(X, Y)〕,〈漢字コード〉〔,〈カラーコード〉〕〔,〈論理演算子〉〕〔,〈モード〉〕

例　PUT KANJI (100, 100), &H3D44, 15　実行結果……(100, 100) に“縦”を表示します。

説　明　漢字ROMがスロットにあるとき、SCREEN 5～8に漢字を表示します。指定できる〈漢字コード〉は、JIS第1水準の漢字コードで、&H2121から&H4F7Eです。

〈論理演算子〉には、AND, OR, XOR, PSET, PRESETの5種類があります。くわしくは、LINEを参照してください。

〈モード〉は0～2で次の意味です。(モード1と2は、二画面交互表示のときに使います。くわしくは、SCREENを参照してください。)

〈モード〉

0	16×16ドットの標準文字表示
1	8×16ドットで偶数ライン表示
2	8×16ドットで奇数ライン表示

漢字ROMが内蔵されていないとき、または漢字ROMカートリッジをつないでいないときは、正常な表示がされないことがあります。SCREEN 7と8は MSX2 のVRAMが128KBの機種でのみ使用できます。

参　照　LINE, COLOR, SCREEN, 資料「漢字ROMコード」

サンプルプログラム

```
10 SCREEN 5
20 KAN=&H3020
30 FOR X=10 TO 240 STEP 20
40 FOR Y=10 TO 190 STEP 20
50 PUT KANJI(X,Y),KAN
60 KAN=KAN+1
70 NEXT Y,X
80 GOTO 80
```

PUT SPRITE

put sprite （プットスプライト：スプライトを置く） ステートメント

働 き スプライトパターンを表示します。

書き方

PUT SPRITE 〈スプライト面番号〉〔, | (X, Y) / STEP (X, Y) | 〕〔, 〈カラーコード〉〕〔, 〈パターン番号〉〕

例 PUT SPRITE 1, (50, 50), 8, 2

説 明 〈パターン番号〉で指定したスプライトパターンを、〈スプライト面番号〉で指定したスプライト面に表示します。

〈スプライト面番号〉はスプライト面の番号で、0～31の値を持つ数式で指定します。

（X, Y）はスプライトパターンを表示する座標を指定します。STEP(X, Y）を付けると、LP（最終参照点）からの相対指定となります。

Xは－32～255（SCREEN 6と7では－32～511：MSX2のみ）の範囲の数式で指定します。また、Yは－32～191（SCREEN 5以降では、－32～211：MSX2のみ）の範囲の数式で指定します。

Y座標に208（SCREEN 4以降では、216：MSX2のみ）を指定するとそのスプライト面以降のスプライトパターンはすべて画面から消えます。

Y座標に209（SCREEN 4以降では217：MSX2のみ）を指定すると指定したスプライト面に表示されていたスプライトパターンが消えます。

〈カラーコード〉はスプライトパターンの色を指定します。

〈スプライトパターン番号〉はSPRITE$システム変数にあらかじめ定義しておいたスプライトパターンの番号で、スプライトパターンのサイズを8×8にしていれば0～255の範囲の数式で指定し、スプライトパターンのサイズを16×16にしていれば0～63の範囲の数式で指定します。

スプライトパターンのサイズはSCREEN文で指定します。

PUT SPRITE文のX、Y座標を省略すると、スプライト面の現在の値が用いられます。また〈カラーコード〉を省略すると前景色となります。

1つのスプライト面には1つのスプライトパターンしか表示できません。このため、以前に同じスプライト面へスプライトパターンを表示していても、それは消去されます。

画面に同時に32個のスプライトパターンの表示を設定することができますが、スプライトパターンが水平方向に5個以上並ぶと、4個までは表示されますが、5つ目以降は画面から消えます。（SCREEN4以降では、9個以上並んだとき、9つ目から消えます。：MSX2のみ）

また、番号の小さいスプライト面ほど優先順位が高いため、同じ座標に表示される複数のスプライトパターンがあれば、番号の小さいスプライト面に表示されているスプライトパターンが手前に見えます。

また、SCREEN 4以降の各スプライトは、1ラインごとに色をつけることができます。詳しくは、COLOR SPRITEを参照してください。

SCREEN 4～8は、MSX2 のみで使用できます。また、SCREEN 7と8は、VRAMが128KBの機種でのみ使用できます。

参 照 SPRITE$, SCREEN, COLOR, COLOR SPRITE, 第1章「スプライト機能」

サンプルプログラム

ペンの形のスプライトを、カーソルで移動します。

```
10 ' *** PUT SPRITE ***
20 COLOR 15,1,1:SCREEN 2,2
30 FOR S=1 TO 2:A$=""
40 FOR P=1 TO 32:READ D$
50 A$=A$+CHR$(VAL("&H"+D$)):NEXT
60 SPRITE$(S)=A$:NEXT
70 DATA 1,2,4,D,17,13,21,23,47,4C,F0
80 DATA C0,0,0,0,0,3F,7E,FC,F8,F0,E0
90 DATA C0,80,0,0,0,0,0,0,0,0
100 DATA 0,0,0,0,0,0,0,0,1,2,4,9,13,27
110 DATA 4F,9F,0,0,0,0,10,28,4C,9E,3F
120 DATA 7E,FC,F8,F0,E0,C0,80
130 A=STICK(0)
140 IF A=1 THEN Y=Y-1
150 IF A=3 THEN X=X+1
160 IF A=5 THEN Y=Y+1
170 IF A=7 THEN X=X-1
180 PUT SPRITE 1,(X,Y+16),15,1
190 PUT SPRITE 2,(X+8, Y),15,2
200 PSET(X-2,Y+29),4
210 GOTO 130
```

READ

read （リード：読む）　ステートメント

働　き DATA文中に書かれているデータを読込み、変数に代入します。

書き方 READ 変数〔,〈変数〉…〕

例 READ A，B，C$

説　明 DATA文中にあるデータを、その順にREAD文中の変数に代入します。

READ文の変数は数値変数でも文字変数でもかまいませんが、読み出された値と変数の型は一致していなければなりません。もし型が一致しない場合には、“Syntax error”が起こります。

〈変数〉を1つ以上指定する場合はカンマ(,)で区切って並べます。

プログラム中のDATA文は、一連のデータとみなされます。読込むDATA文を指定するときはRESTORE文を使います。

〈変数〉の数がDATA文のデータの個数を越えてしまった場合は、“Out of DATA” のエラーメッセージが表示されます。DATAの余分なデータは無視されます。

参　照 RESTORE，DATA

サンプルプログラム

```
10 '** READ **
20 READ A$,B$:PRINT A$;B$
30 DATA ｺﾚﾊ,ｱﾏﾘ
40 RESTORE 100:READ A$,B$
50 PRINT A$;B$
60 RESTORE 80
70 READ A$:PRINT A$
80 DATA ｱﾘﾏｾﾝ
90 END
100 DATA ﾖｲ,ﾚｲﾃﾞﾊ
```

REM

remark（リマーク：注目、注意）　ステートメント

働　き　プログラムに注釈文を入れます。

書き方　REM〔〈注釈文〉〕

例　REM　ニュウリョク A，B
　　'ニュウリョク A，B

説　明　REM文から、行の終りまでは注釈文とみなされ、実行時には無視されます。REM文の後にコロン(：)で区切って実行文を書いても実行されません。

REM文では予約語の“REM”の代わりにアポストロフィ(')を使うことができますが、リストアウトしてもREMにはなりません。

サンプルプログラム

```
10 '** REM  **
20 'REM ﾊ (') ﾃﾞﾓ ｵﾅｼﾞ ﾃﾞｽ｡
30 REM PRINT"ﾌﾟﾘﾝﾄ ｻﾚﾏｾﾝ"
40 END
```

RENUM

renumber（リナンバー：番号のつけ直し）　コマンド

働　き　プログラムの行番号をつけ直します。

書き方　RENUM〔〈新行番号〉〕〔，〈旧行番号〉〕〔，〈増分〉〕

例　RENUM

説　明　〈新行番号〉は、新しくつける行番号の最初の行番号で、省略値は10です。

〈旧行番号〉は行番号のつけ替えをはじめる現在のプログラムの行番号です。省略値はそのプログラムの最初の行番号です。

〈増分〉は新しく付ける各行番号の間隔で、省略値は10です。

この命令によりGOTO，GOSUB，THEN，ON～GOTO，ON～GOSUBおよびERL文などの行番号も合わせて変更します。ただし、これらの文が参照している行番号の行が存在しない場合には、“Undefined line xxxxx in yyyyy”というエラーメッセージが表示されます。この場合、誤った行番号（xxxxx）は変更されません。

- 65529以上の行番号を発生するような場合には“Illegal function call”エラーとなります。
- プログラムの順序が変化する場合もエラーとなります。

例　10　GOTO30
　　20　END　→　RENUM　15，30を行なうとエラーになります。
　　30　GOTO20　（30行が10行と20行の間に入ってしまうため）

R

RESTORE

restore（リストア：用意し直す）　ステートメント

働き　READ文で読込むDATA文の開始行を指定します。

書き方　RESTORE〔〈行番号〉〕

例　RESTORE

説明　同じDATA文を複数回にわたり読込んだり、DATA文を指定するときに実行します。

〈行番号〉を省略すると、次に実行するREAD文はプログラム中の最初のDATA文のデータから読み始めます。なお、〈行番号〉には、変数、計算式は指定できません。

参照　READ

RESUME

resume（リジューム：再び始める）　ステートメント

働き　エラー処理を終了し、プログラムの実行を再開します。

書き方

```
RESUME〔 | 0        | 〕
         | NEXT     |
         | 〈行番号〉 |
```

例

```
RESUME  0
RESUME  NEXT
RESUME  50
```

説明　エラー処理プログラムの実行後、メインプログラムの実行を再開します。

プログラムの実行を再開する場所に応じて3つの書式を選ぶことができます。

書式	説明
RESUME RESUME　0	エラーの発生した文から実行が再開されます。
RESUME　NEXT	エラーの発生した文の次の文から実行が再開されます。
RESUME〈行番号〉	〈行番号〉で指定した行から実行が再開されます。

参照　ON　ERROR　GOTO

RETURN

return（リターン：もどる）　ステートメント

働　き サブルーチンから復帰します。

書き方 RETURN〔〈行番号〉〕

例　RETURN
RETURN 500

説　明 RETURN文はGOSUB文によりジャンプしたサブルーチンを終了し、GOSUB文の次の文へ復帰（実行を開始）します。また、割込み処理ルーチンで使うと、割込み処理を終了し、割込みがかかったときに中断されたプログラムが再開されます。

〈行番号〉を指定すると、サブルーチンや割込み処理ルーチン終了後、〈行番号〉へジャンプします。

サブルーチンや割込み処理ルーチンからの復帰には必ずRETURN文を使ってください。また、1つのサブルーチン中に複数個のRETURNがあってもかまいませんが、GOSUB文と正しく対応していなければなりません。

サブルーチン中でCLEAR文を実行すると、RETURN文による復帰ができなくなり"RETURN without GOSUB" エラーとなります。

参　照 GOSUB, ON GOSUB, 第1章「割込み」

RIGHT$

right$（ライトドル：文字列の右側）　関数

働　き 文字列の右側から指定した長さの文字列を得ます。

書き方 RIGHT$(〈文字列〉,〈数〉)

例　PRINT RIGHT$(A$, 4)

説　明 〈文字列〉の右側から〈数〉で指定された数の文字の文字列を得ます。〈数式〉の範囲は0～255です。
〈数式〉の値が0ならば、RIGHT$はヌルストリングを得ます。
〈数式〉が〈文字列〉の文字数より大きければ、〈文字列〉のすべてを値とします。

参　照 LEFT$, MID$

R

サンプルプログラム

```
10 ' ** RIGHT$ **
20 A$="RIGHT$ﾊ ﾓｼﾞﾚﾂ ﾉ ﾐｷﾞｶﾗ ﾄﾘﾀﾞｼﾏｽ"
30 L=LEN(A$)
40 FOR R=1 TO L
50 B$=RIGHT$(A$,R)
60 PRINT B$
70 NEXT
```

RND

random（ランダム：適当な）　関数

働き　0から1の間の乱数を得ます。

書き方　RND〔(〈数式〉)〕

例　R=RND (1)

説明　0より大きく1より小さい乱数を得ます。発生する乱数は、RUN文が実行されるごとに同系列です。

発生される乱数は〈数式〉の値によって次のように異なります。
〈数式〉が負の場合——乱数系列を初期化します。
〈数式〉が0の場合——1つまえに発生した乱数の値をとります。
〈数式〉が正の場合——次の乱数を得ます。

RND(−TIME) を使って毎回ふり出す乱数の並びを変えることができます。

サンプルプログラム

```
10 COLOR ,1,1:SCREEN 3
20 PSET(RND(1)*255,RND(1)*191),RND(1)*15
30 GOTO 20
```

RUN

run（ラン：動く）　コマンド

働き　メモリにあるベーシックのプログラムを実行する。

書き方　RUN〔〈行番号〉〕

例　RUN 100

説明　プログラムを〈行番号〉から実行します。〈行番号〉の指定のない場合には、最も若い行番号から実行がはじまります。
プログラムを実行する前に、開かれているファイルはすべて閉じられます。

SAVE

save（セーブ：与える）　　コマンド/Disk BASIC

働　き　メモリのBASICプログラムをファイルに記録（セーブ）します。

書き方　SAVE "<ファイルスペック>"〔, A〕

例　SAVE "CAS：DEMO"

説　明　<ファイルスペック>で指定するファイルにプログラムを書き込みます。

カセットレコーダにはアスキー形式でセーブされますが、フロッピーディスクにはバイナリ形式でセーブされます。フロッピーディスクにアスキー形式でセーブする場合は<ファイルスペック>の後にアスキーオプション（, A）をつけます。

アスキー形式でプログラムをセーブしておくと、プログラムはCLOAD文で作成するバイナリ形式ファイルのように圧縮されず、アスキーキャラクタ形式のままファイルに保存されます。

アスキー形式のファイルはバイナリ形式のファイルよりも多くのファイルスペースを必要としますが、MERGEコマンド（プログラムの混合）を使うことができます。また、LINE INPUT文によりデータファイルとして読み出すこともできます。

アスキー形式のファイルにはプログラムの各行改行コード（LINE INPUT参照）で区切られてセーブされます。
アスキー形式のファイルの最後には終了マークとして CTRL + Z コード（CHR$(26)）がついています。

アスキー形式でセーブしたプログラムは、LOAD命令、MERGE命令でロードします。

参　考　LOAD, MERGE

R
S

SCREEN

screen（スクリーン：画面）　　ステートメント

働き 画面のモード、スプライトサイズ、キークリック音の有無、カセットボーレート、専用プリンタの有無を指定します。

書き方 SCREEN〔〈画面モード〉〕〔,〈スプライトサイズ〉〕〔,〈キークリック〉〕〔,〈ボーレート〉〕〔,〈プリンタオプション〉〕〔,〈インターレスモード〉〕

例 SCREEN 1，2

説明 SCREEN文を実行すると、今まで表示されていた画面が消え、新しい画面モードになります。

●画面の指定：〈画面モード〉は0～8の値を持つ数で指定します。（SCREEN 4～8は MSX2のみ）

0：最大40×24文字で画面が構成されるテキストモード。（MSX2では80×24まで可能）
1：最大32×24文字で画面が構成されるテキストモード。
2：256×192ドットで画面が構成される高解像度グラフィックモード。
3：64×48ブロックで画面が構成されるマルチカラーモード。1ブロックは、4×4ドットです。
4：256×192ドットで画面を構成します。SCREEN2のスプライト機能を拡張しています。
5：256×212ドットで画面を構成します。1点ごとに色を設定できます。
（512色中 16色使用可）
6：512×212ドットで画面を構成します。1点ごとに色を設定できます。
（512色中 4色使用可）
7：512×212ドットで画面を構成します。1点ごとに色を設定できます。
（512色中 16色使用可）
8：256×212ドットで画面を構成します。1点ごとに色を設定できます。
（256色を同時に使用可）

テキストモードではプログラムやメッセージなどが表示できますが、グラフィック命令(LINE文やCIRCLE文など）は使えません。逆に、グラフィックモードでは、プログラムやメッセージなどは表示されませんが、グラフィック命令を使って絵を描くことができます。
また、画面モード0の場合に限ってスプライト機能を使うことはできません。
なお、INPUT文を実行したりエラーが発生した場合には自動的にテキストモードになります。

●スプライトの大きさ：〈スプライトサイズ〉は0～3の値を持つ式でスプライトパターンの大きさを指定します。以後、SPRITE$でスプライトパターンを定義するときは、ここで指定した構成に従います。

0：スプライトを8×8ドットで構成する。
1：スプライトを8×8ドットで構成し、縦横それぞれ2倍に拡大して表示する。
2：スプライトを16×16ドットで構成する。
3：スプライトを16×16ドットで構成し、縦横それぞれ2倍に拡大して表示する。

●キークリック音：〈キークリック〉はキーを押したときにクリック音をだすかどうかを設定します。〈キークリック〉は0または1の値を持つ数で指定します。

0：キークリック音を出さない。
1：キークリック音を出す。

●カセットのボーレート：〈ボーレート〉はカセットテープへプログラムやデータを書き出すときのボーレート（書き出す速さ）を指定します。CSAVE，BSAVEコマンドでボーレートを省略すると、ここで、指定したボーレートで書き出されます。

1：1200ボー
2：2400ボー

カセットテープからの読込み時は、ボーレートは自動的に選択されます。

●プリンタの設定：〈プリンタオプション〉は、使用しているプリンタがMSX標準のプリンタかどうかを指定します。
MSX標準でないプリンタでは、グラフィック文字は空白に、また、ひらがなはカナ文字に置換えられます。〈プリンタオプション〉は次のような値を持つ式で指定します。

0　　：MSX標準のプリンタ
0以外：MSX標準以外のプリンタ

●インターレスモード：MSX2では、〈インターレスモード〉を設定して、画面のインターレス走査および2つの画面の交互表示ができます。

〈インターレスモード〉に0～3を指定します。省略値は0です。

0：通常表示（ノン　インターレス）
1：インターレス表示（ディスプレイページのみ）
2：ノン・インターレス表示・二画面交互表示
3：インターレス表示・二画面交互表示

インターレス表示は、パソコン専用の高解像度テレビや特殊な表示をさせたいときに使います。2～3を指定するときは、ディスプレイページが奇数である必要があり、ディスプレイページとそれより1つ小さい番号のページが交互に表示されます。

例

別のページに書いた線を重ねます。

```
10 COLOR 15,1:SCREEN 6,,,,,0
20 SET PAGE 0,0:CLS
30 LINE(0,0)-(511,211)
40 FOR K=0 TO 500:NEXT
50 SET PAGE 1,1:CLS
60 LINE(0,211)-(511,0)
70 FOR K=0 TO 500:NEXT
80 SCREEN ,,,,,2
90 GOTO 90
```

参照 WIDTH、第1章「画面モード」，SET PAGE，COLOR，COLOR SPRITE

S

SET

set（セット：設定する） MSX2

MSX2では、電源スイッチ“入”のときに自動設定したい様々な項目を、パソコン内のメモリに設定できます。
MSX2のパソコンには、CMOS（シー・モス）というメモリを内蔵しています。CMOSは電池などでバッテリーバックアップされているので、電源スイッチを切にしても内容が消えません。セットできるものは、次の8つです。

SET ADJUST …………テレビ画面の表示位置の調整
SET BEEP ………………BEEP音（エラーしたときの音）の設定
SET DATE ………………日付の設定
SET PASSWORD ……パスワードの設定
SET PROMPT …………プロンプト（通常“OK”）の設定
SET SCREEN …………SCREEN，WIDTHの設定
SET TIME……………………時刻の設定
SET TITLE ………………初期画面のタイトルの設定

注意！ SET PASSWORD，SET PROMPT，SET TITLEの文字の機能は同時に使うことができません。2つ以上を指定すると、あとで指定したものが設定として残ります。

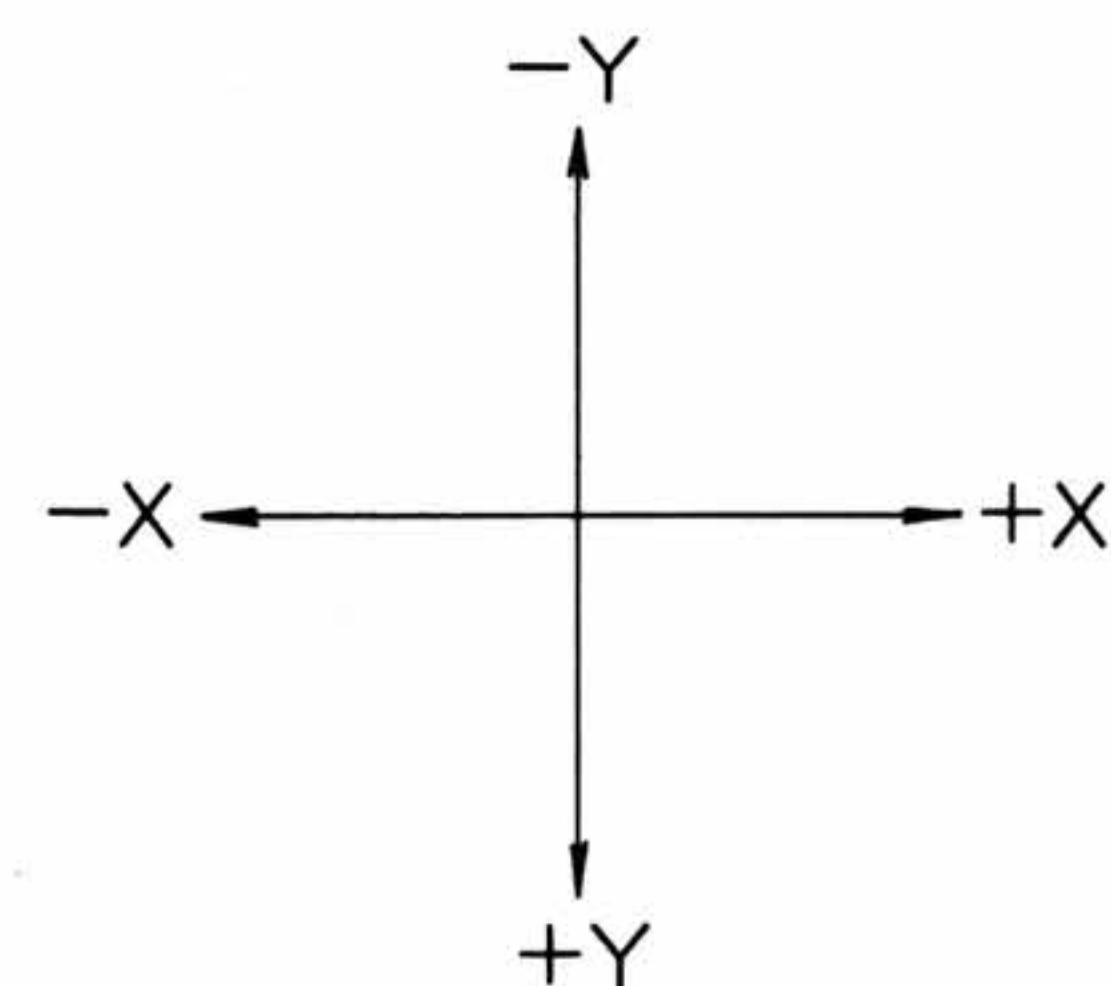
set adjust（セット・アジャスト：差を調整する） MSX2

説 明 テレビ画面の表示位置を上下左右にずらします。

書き方 SET ADJUST（X座標、Y座標）

例 SET ADJUST（5，3）

説 明 画面の表示位置を上下左右にずらすことができます。指定できる値は、X，Yともに−7〜+8です。
例では右に5ドット、下へ3ドット ずらします。

−Y
−X +X
+Y

set beep（セット・ビープ：ビー音の設定） MSX2

働　き　BEEP音を設定します。

書き方　SET BEEP〈音色〉,〈音の大きさ〉

例　SET BEEP 3, 3　　実行結果……エラーのとき〝ピンポン〞と音が出ます。

説　明　〈音色〉と〈音の大きさ〉はそれぞれ1～4の値です。
〈音色〉の1は、MSX と同じです。
〈音の大きさ〉は、4で最大となります。
〈音色〉および、〈音の大きさ〉の一方を省略することはできますが、二つを省略することはできません。
例. SET BEEP 4　　SET BEEP, 3

set date（セット・デイト：日付の設定） MSX2

働　き　日付を設定します。

書き方　SET DATE〝〈年/月/日〉〞〔,A〕

例　SET DATE〝85/01/01〞　　実行結果……現在の日付を1985年1月1日とします。

説　明　パソコン内の時計の日付を設定します。
日付は例のように引用符（"）で囲み、年、月、日の間をスラッシュ（/）で区切ります。月日が1桁のときは、0を前につけて2桁としてください。

,Aのオプションをつけると、アラームの日付を設定します。
アラーム機能については各機種で異なりますのでパソコンの取扱説明書をご覧ください。

S

set password（セット・パスワード：パスワードの設定） MSX2

働　き　電源〝入〞のときパスワードをキー入力しなければパソコンが使えないようにします。

書き方　SET PASSWORD〝〈文字列〉〞

例　SET PASSWORD〝MSX 2〞

説　明　パスワード（合い言葉）を設定すると電源〝入〞のとき初期画面にPassword:と表示されます。設定した文字列をキー入力して リターン キーを押すまで初期画面のままです。

〈文字列〉は255文字以内で、引用符（"）で囲んで設定します。

設定したパスワードを忘れてしまったときは、GRAPH キーと STOP キーを押しながら電源を〝入〞にしてください（初期画面が変わるまで押し続けてください。）

set prompt（セット・プロンプト：注意メッセージの設定） MSX2

働　き　プロンプト（"OK" など）を設定します。

書き方　SET PROMPT "〈プロンプト文〉"

例　SET PROMPT "Ready"　　実行結果……OKの替りにReadyと表示されます。

説　明　BASICのプロンプト "OK" を他の文字に変えることができます。
〈プロンプト文〉は6文字以内の文字列です。

set screen（セット・スクリーン：画面の設定） MSX2

働　き　SCREENの設定を初期状態として設定します。

書き方　SET SCREEN

説　明　現在設定されているSCREEN文とWIDTH文の設定を記憶し、次に電源 "入" としたときの初期設定値とします。
設定内容：画面モードと表示幅（テキストモードと、その表示幅）、前景色、背景色、周辺色、キースイッチ、キークリック音、プリンタオプション、カセットボーレイト、ディスプレイモード

set time（セット・タイム：時刻の設定） MSX2

働　き　時刻を設定します。

書き方　SET TIME "〈時：分：秒〉"〔, A〕

例　SET TIME "12：34：00"　　実行結果……現在の時刻を12時34分0秒とします。

説　明　パソコン内の時計の時刻を設定します。
時刻は例のように引用符（〃）で囲み、時、分、秒の間をコロン（：）で区切ります。
時刻が1桁（0～9秒など）のときは、0を前につけて2桁（00～09）としてください。

, Aのオプションをつけると、アラームの時刻を設定します。アラーム機能については各機種で異なりますので、パソコンの取扱説明書をご覧ください。

set title（セット・タイトル：タイトルの設定） MSX2

働き 電源"入"のときの画面の色と、タイトル文字を設定します。

書き方 SET TITLE "〈タイトル文字〉"〔,〈色番号〉〕

例 SET TITLE "ベーシック"

説明 電源を"入"としたとき、およびリセットしたときに表示される初期画面の色と、〈タイトル文字〉を指定します。
〈タイトル文字〉は6文字以内の文字列です。例のようにちょうど6文字のときは、何かキーを押すまで初期画面が表示され続けます。
〈色番号〉は1～4で指定します。色と番号は次のように対応しています。

1：青
2：緑
3：赤
4：オレンジ

SET PAGE

set page（セット・ページ：） MSX2

働き VRAMを複数のページに分割し、表示するページとグラフィック命令で書き込むページを設定します。

書き方 SET PAGE〔〈ディスプレイページ〉〕〔,〈アクティブページ〉〕

例 SET PAGE 0，1

説明 MSX2ではVRAMが64KBまたは128KBありますが、各SCREENでの画面表示に全部のVRAMを使っているわけではありません。SET PAGE命令はVRAMを複数の画面（ページといいます）に分割し、複数の画面を交互に表示させることができます。
この機能は画面モードがSCREEN 5～8のとき有効で、各画面モードで指定できるページの数は次のとおりです。

〈画面モード〉	〈VRAM64KBの機種〉	〈VRAM128KBの機種〉
SCREEN 5	0～1ページ	0～3ページ
SCREEN 6	0～1ページ	0～3ページ
SCREEN 7	使用不可	0～1ページ
SCREEN 8	使用不可	0～1ページ

（SCREEN 7と8はVRAM容量が128KBの機種のみ使用できます）

S

SET PAGE

MSX2

〈ディスプレイページ〉とは、実際に画面に表示されるページです。

〈アクティブページ〉とは、グラフィック命令等でデータが書き込まれるページです。SET PAGE命令でページを切換えても、そのページの内容はクリアされません。ページを切換えて書き込むときは、CLS命令で前の画面をクリアしてください。

〈ディスプレイページ〉および〈アクティブページ〉の一方を省略することができます。このときは以前に指定したページが指定されます（初期値は0です）

サンプルプログラム1

```
10 SCREEN 5
20 SET PAGE 0,0:CLS
30 LINE(10,10)-(211,200),3,B
40 SET PAGE 1,1:CLS
50 CIRCLE(127,105),80,8
60 K=0
70 SET PAGE K:K=1-K
80 FOR V=0 TO 500:NEXT
90 GOTO 70
```

サンプルプログラム2

```
10 SCREEN 5
20 FOR D=1 TO 5:READ A(D),C(D)
30 TA=TA+A(D):NEXT D
40 SET PAGE 0,0:CLS
50 LINE(10,10)-(250,201),15,B
60 FOR D=1 TO 5
70 LINE(D*30+20,200)-STEP(20,-A(D)*3),C(
D),BF
80 NEXT D
90 R=6.2831/TA:S2=1E-04
100 SET PAGE 1,1:CLS
110 LINE(10,10)-(250,201),15,B
120 FOR D=1 TO 5
130 S1=S2:S2=S2+R*A(D):S=(S1+S2)/2
140 CIRCLE(127,105),80,C(D),-1*S1,-1*S2
150 PAINT STEP(COS(S)*5,-SIN(S)*5),C(D)
160 NEXT:K=0
170 SET PAGE K:K=1-K
180 FOR T=1 TO 1000:NEXT
190 GOTO 170
200 DATA 40,3,30,9,20,7,35,11,25,15
```

ちょっと一言

VRAM容量が128KBのとき、サンプルプログラム2の170行〜190行を右のように変えてみてください。クリアされないのがわかりますよ。

```
170 SCREEN 7
180 SET PAGE 1
190 GOTO 190
```

SET VIDEO

set video（セット・ビデオ）　MSX2

働き スーパーインポーズ、オーディオミキシングなどを設定します。

書き方 SET VIDEO〔〈モード〉〔,〈Ym〉〔,〈CB〉〔,〈同期〉〔,〈音声〉〔,〈ビデオ入力〉〔,〈AVコントロール〉〕〕〕〕〕〕〕

説明 この機能は、MSX2 のオプション機能です（標準機能ではありません。）この機能を備えた機種のみに限り有効です。

〈モード〉は0～3の整数で、スーパーインポーズのモードを設定します。

〈モード〉	〈表示画面〉
0	コンピュータ
1	コンピュータ
2	スーパーインポーズ
3	テレビ（外部信号）

※モードが0のときは外部同期はできません。
また、モード1～3では、コンポジット信号が出力されません。

〈Ym〉が1のとき、テレビの明るさが半分になり、0のとき通常の明るさとなります。
〈CB〉が1のとき、VDPのカラーバスが入力状態となり、0のとき出力状態となります。

〈同期〉が1であれば外部同期に、0であれば内部同期になります。

〈音声〉は、外部音声信号を混合して出力するかを設定します。

〈音声〉	〈機能〉
0	外部音声信号を混合しない。
1	右チャンネルの外部音声信号とパソコンの音声信号を混合する
2	左チャンネルの外部音声信号とパソコンの音声信号を混合する
3	両チャンネルの外部音声信号とパソコンの音声信号を混合する

〈ビデオ入力〉は外部映像信号の入力を切換えます。0であればRGBマルチ端子からの信号が、1のときビデオ入力端子からの信号を選択します。

〈AVコントロール〉は、RGBマルチ端子のAVコントロール出力信号を0のときOFFに、1のときONとします。

SGN

sign（サイン：符号）　　関数

働　き 正負の符号を調べます。

書き方 SGN（〈数〉）

　　例　　PRINT SGN（10）　　　実行結果……1

説　明 〈数〉が正の場合は1を、〈数〉が0の場合は0を、〈数〉が負の場合には－1を結果として得ます。関数のため、他の命令と組み合せて使用します。

サンプルプログラム

```
10 PRINT "   A   INT(A) SGN(A)"
20 FOR A=-2.5 TO 2.5 STEP .6
30 B=INT(A):C=SGN(A)
40 PRINT USING"##.#   ##.#   ##.# ";A;B;C
50 NEXT
```

SIN

sin（サイン：正弦）　　関数

働　き 正弦（サイン）を得ます。

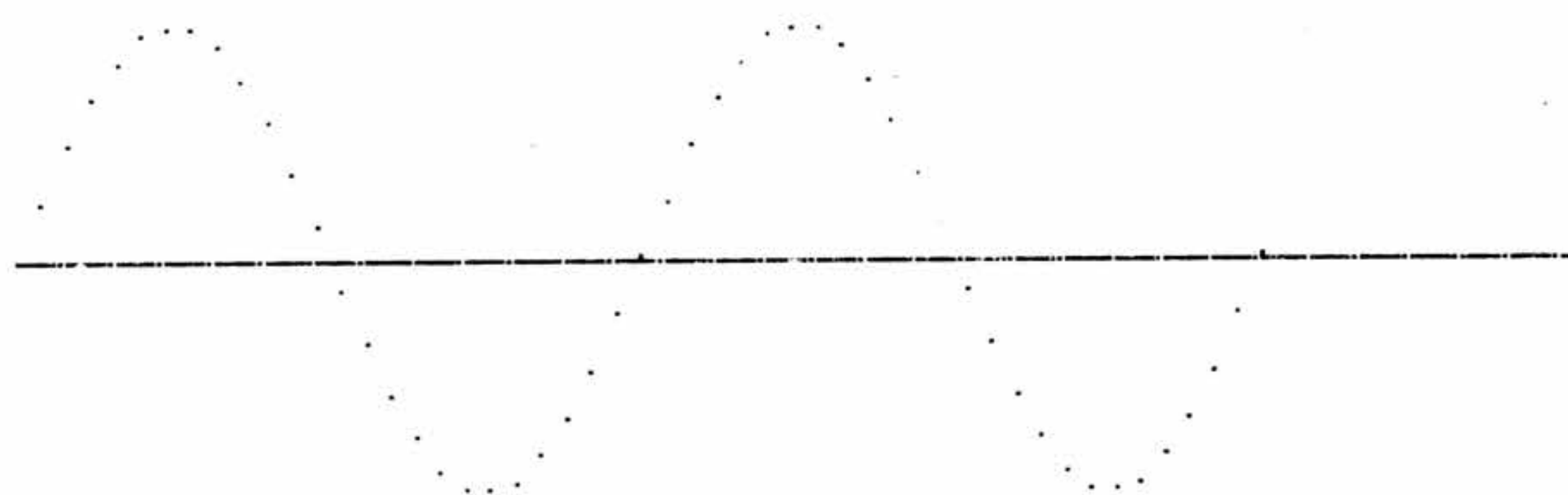
書き方 SIN（〈数〉）

　　例　　PRINT SIN（1）　　　実行結果…….84147098480792

説　明 〈数〉の正弦（サイン）を求めます。〈数〉の単位はラジアン（CIRCLE参照）です。〈数〉は整数型、単精度型、倍精度型のいずれでもかまいませんが、結果は倍精度で得られます。関数のため、他の命令と組み合せて使用します。

参　照 COS，TAN

サンプルプログラム

```
10 SCREEN 2
20 LINE(0,100)-(250,100)
30 FOR X=0 TO 200 STEP 4
40 R=3.1416/50
50 Y=100-40*SIN(R*X)
60 PSET(X,Y)
70 NEXT
80 GOTO 80
```

sound（サウンド：音）　ステートメント

働き PSG（サウンドIC）のレジスタにデータを書き込み、音を発生させます。

書き方 SOUND 〈レジスタ番号〉，〈データ〉

例 SOUND 0，24：SOUND 7，& HFE：SOUND 8，10

説明 PSG（プログラマブルサウンドジェネレータ）のレジスタ（R0～R13の14個）に直接データを書き込んで音を出します。PLAY文のように簡単に音を出すわけには行きませんが、細い指定をすることができるので、複雑な音やゲームなどの効果音を出すことができます。

PSGの音の発生装置と音量の制御装置は3チャンネルに独立していますので、別々に指定することができます。ただし、ノイズの発生装置とエンベロープの発生装置は各チャンネルで共用します。

R0，R1	チャンネルAの音の周波数を指定します。
R2，R3	チャンネルBの音の周波数を指定します。
R4，R5	チャンネルCの音の周波数を指定します。
R6	ノイズ周波数を指定します。
R7	各チャンネルから音を出すかどうかを指定します。
R8，R9，R10	チャンネルA，B，Cのそれぞれの音量を指定します。 L0～L3を0にすると無音に、15にすると最大の音量になります。エンベロープの指定をするときは、Mを1にします。
R11，R12	エンベロープの周期を2つのレジスタを使って指定します。
R13	エンベロープ形状を指定します。

●周波数の設定（R0～R5）

チャンネルAのR0，R1を例に説明します。

出力したい周波数をf_Tとすると、

$$\underset{(上位1バイト)}{FT} + \underset{(下位1バイト)}{CT} = \frac{1.78977 \times 10^6}{16 \times f_T}$$

CTをR0に、FTをR1に書き込みます。

例） f_T＝400Hzの場合

$$\frac{1.78977 \times 10^6}{16 \times 400} \fallingdotseq 279$$

279は&H117ですから、R1に&H1　R0に&H17を設定します。

SOUND R1，&H1
SOUND R0，&H17

●ノイズ周波数の設定（R6）

出力したいノイズ周波数をf_Nとすると

$$NP = \frac{1.78977 \times 10^6}{16 \times f_N}$$

NPをR6に書き込みます。

レジスタ ＼ ビット		B7	B6	B5	B4	B3	B2	B1	B0
R0	チャンネルA周波数	8BIT FT A							
R1						4BIT CT A			
R2	チャンネルB周波数	8BIT FT B							
R3						4BIT CT B			
R4	チャンネルC周波数	8BIT FT C							
R5						4BIT CT C			
R6	ノイズ周波数				5BIT NP				
R7	チャンネル設定			NOISE			TONE		
				C	B	A	C	B	A
R8	チャンネルA音量				M	L3	L2	L1	L0
R9	チャンネルB音量				M	L3	L2	L1	L0
R10	チャンネルC音量				M	L3	L2	L1	L0
R11	エンベロープ周期	8BIT FT							
R12		8BIT CT							
R13	エンベロープ周期					E3	E2	E1	E0

S

例） f_N＝8KHzの場合

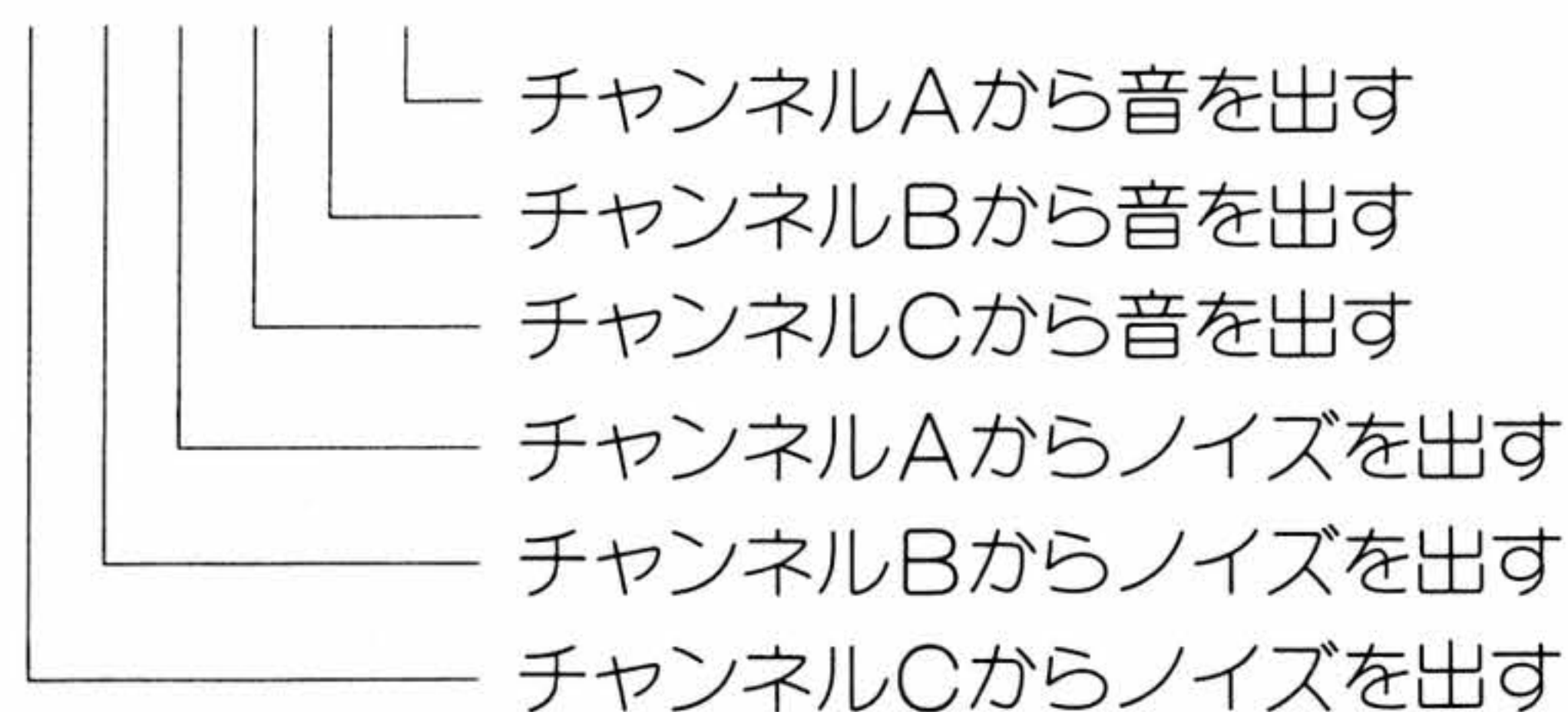
$$\frac{1.78977\times10^6}{16\times8000}\fallingdotseq13$$

従ってR6に13(&HD)を設定します。

SOUND R6，13

● **チャンネルの設定（R7）**

使用したいチャンネルに対応するビットを0にします。

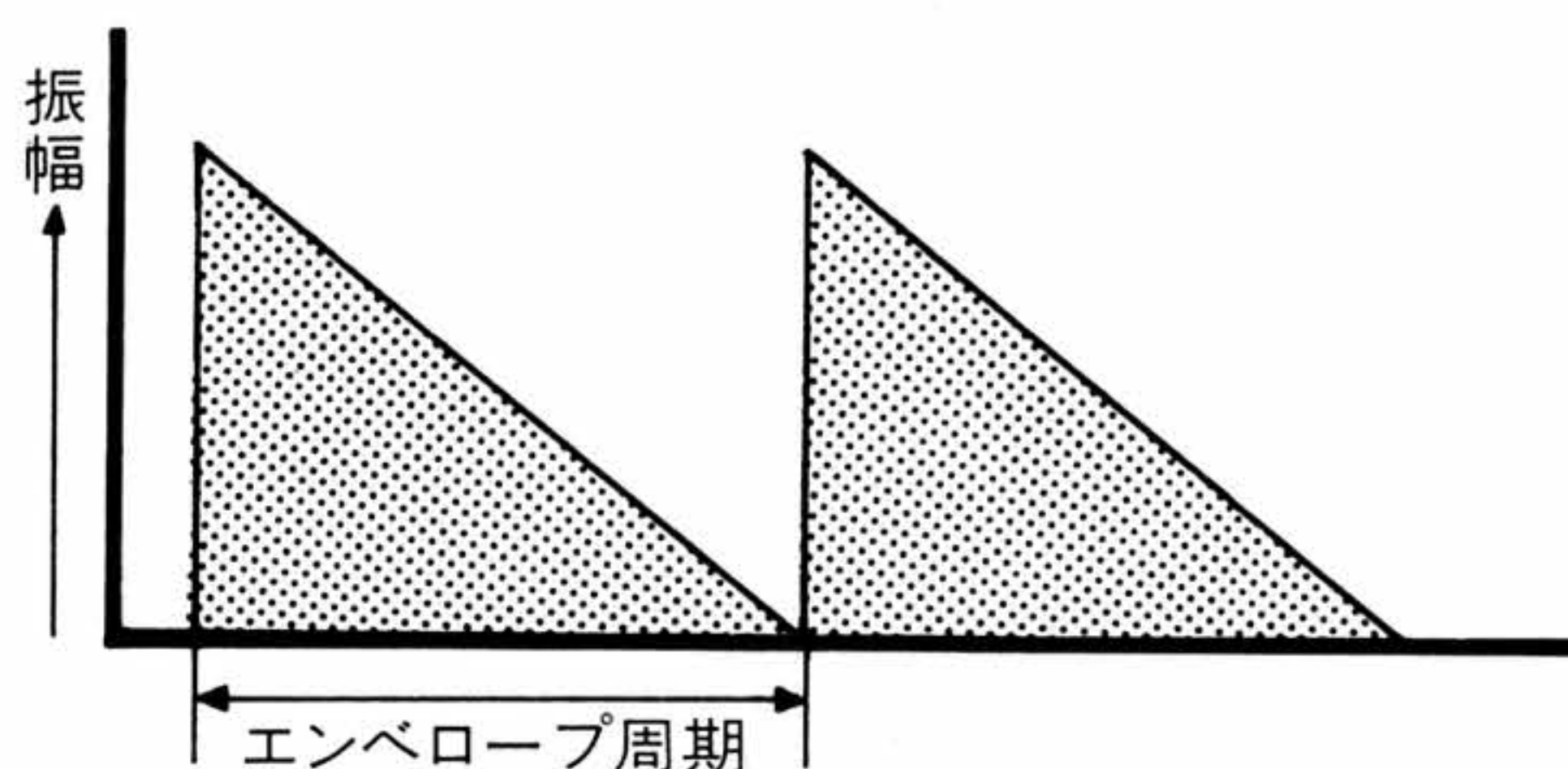
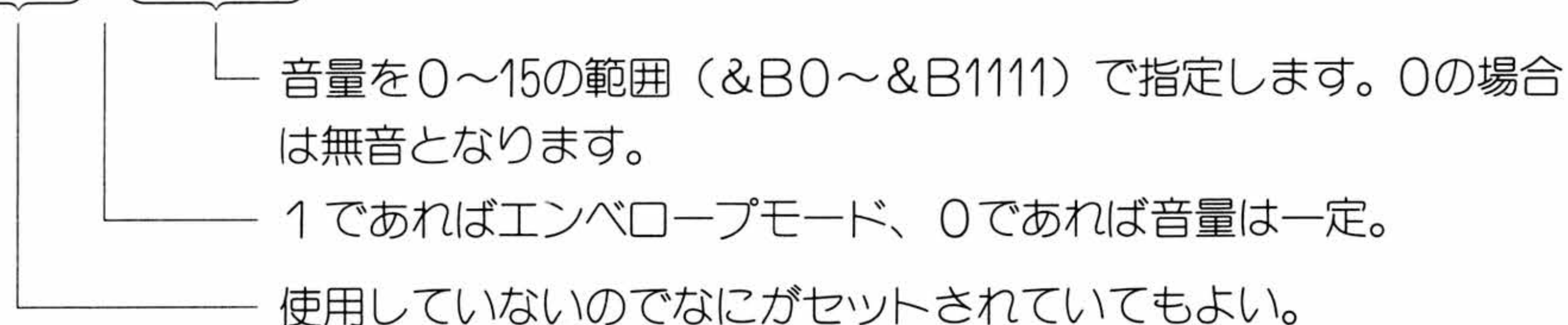
```
××000000
  ││││││
  │││││└─ チャンネルAから音を出す
  ││││└── チャンネルBから音を出す
  │││└─── チャンネルCから音を出す
  ││└──── チャンネルAからノイズを出す
  │└───── チャンネルBからノイズを出す
  └────── チャンネルCからノイズを出す
```

例） チャンネルA，Cから音を出します。（他の動作はしない）

R7＝&B111010

● **音量設定（R8，R9，R10）**

音量設定には大きさが一定のものと、時間的に変化するエンベロープモードとがあります。エンベロープモードを指定している場合はさらにエンベロープの形状とその周期を指定します。

```
&B×××1××××
```

- ××××：音量を0～15の範囲（&B0～&B1111）で指定します。0の場合は無音となります。
- 1：1であればエンベロープモード、0であれば音量は一定。
- ×××：使用していないのでなにがセットされていてもよい。

● **エンベロープ周期の設定（R11，R12）**

音が増加・減衰する時間をT_Eとすると、

エンベロープ周波数 $f_E=1/T_E$

$$FT+CT=\frac{1.78977\times10^6}{256\times f_E}$$

FTをR11，CTをR12に書き込みます。

例） 音が減衰して0になるまでの時間を2秒にする。

$$f_E = 1/2 = 0.5$$

$$\frac{1.78977 \times 10^6}{256 \times 0.5} \fallingdotseq 13983$$

13983は &H369Fですから R11に &H36、
R12に &H9Fを設定します。

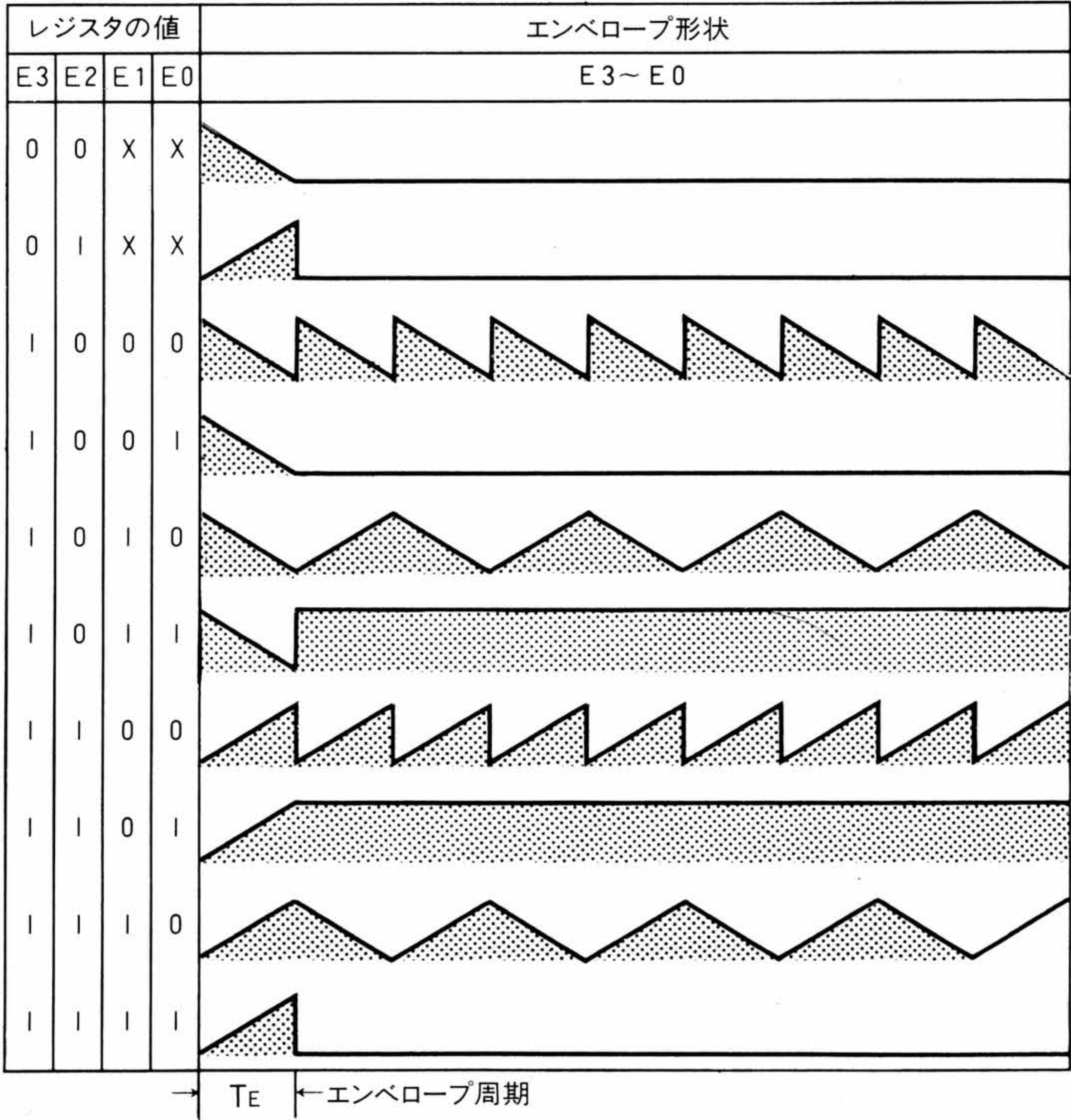
```
SOUND R11, &H36
SOUND R12, &H9F
```

●エンベロープ形状の設定（R13）

エンベロープ形状はレジスタ R13（E0〜E3）へ0〜15（&B0〜&B1111）で指定します。

エンベロープ発生器出力

レジスタの値				エンベロープ形状
E3	E2	E1	E0	E3〜E0
0	0	X	X	
0	1	X	X	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

→ T_E ←エンベロープ周期

SPACE$

space$（スペースドル：空白の文字列）　関数

働　き　指定した長さの空白の文字列を用意します。

書き方　SPACE＄（〈数式〉）

例　PRINT　SPACE＄(10)；"ABC"

説　明　〈数式〉の数だけの空白をもった文字列を与えます。

〈数式〉の値の範囲は0～255ですが、電源投入時は、文字列のメモリ空間が200となっています。このため値は0～200の範囲で設定します。これを変更するにはCLEAR命令を用います。

参　照　SPC，CLEAR，第5章「メモリマップ」

サンプルプログラム

```
10 FOR T=0 TO 20
20 A$="*"+SPACE$(T)+"*"
30 PRINT LEN(A$);
40 PRINT A$
50 NEXT T
```

SPC

space（スペース：空白）　関数

働　き　画面やプリンタに指定した数だけ空白を出力する。

書き方　SPC（〈数式〉）

例　PRINT　SPC(10) A＄

説　明　SPC関数はPRINT文やLPRINT文など出力文中のみ使用することができます。
〈数式〉の値の範囲は0～255です。関数のため、他の命令と組み合せて使用します。

参　照　SPACE＄，TAB

サンプルプログラム

```
10 FOR T=1 TO 10
20 PRINT "*";SPC(T);"*"
30 NEXT T
```

```
run
* *
*  *
*   *
*    *
*     *
*      *
*       *
*        *
*         *
*          *
```

SPRITE ON／OFF／STOP

sprite on　（スプライトオフ：スプライト割込みの許可）
sprite off　（スプライトオフ：スプライト割込みの禁止）　ステートメント
sprite stop（スプライトストップ：スプライト割込みの保留）

働　き　スプライトパターンが重なったときに発生する割込みを許可、禁止、または保留します。

書き方　SPRITE　ON
SPRITE　OFF
SPRITE　STOP

説　明　SPRITE　ON文はスプライト割込みを許可します。この命令を実行しておくと、PUT SPRITE文で画面にスプライトパターンを表示したとき、他のスプライトパターンと重なっていれば割込みが発生し、実行中のプログラムを中断してON SPRITE GOSUB文で定義しておいた割込み処理ルーチンが実行されます。

SPRITE　OFF文はスプライト割込みを禁止します。この命令を実行しておくと、スプライトパターンが重っていても割込みはかかりません。

SPRITE　STOP文はスプライト割込みを保留します。この命令を実行しておくと、以後にかかったスプライト割込みはSPRITE　ONを実行するまで保留され、SPRITE　ON文を実行するとすぐに割込み処理ルーチンが実行されます。

SCREEN4以降の画面で、COLOR　SPRITEによって特殊なスプライトカラーを指定しておくと、その色の部分には割込みが発生しません。くわしくはCOLOR SPRITEを参照してください。（MSX2のみ）

参　照　ON SPRITE GOSUB，PUT SPRITE，COLOR SPRITE

S

SPRITE$

sprite$（スプライトドル：スプライトパターンの作成）　システム変数

働　き　スプライトのパターンを作ります。

書き方　SPRITE$（〈スプライトパターン番号〉）=〈文字式〉

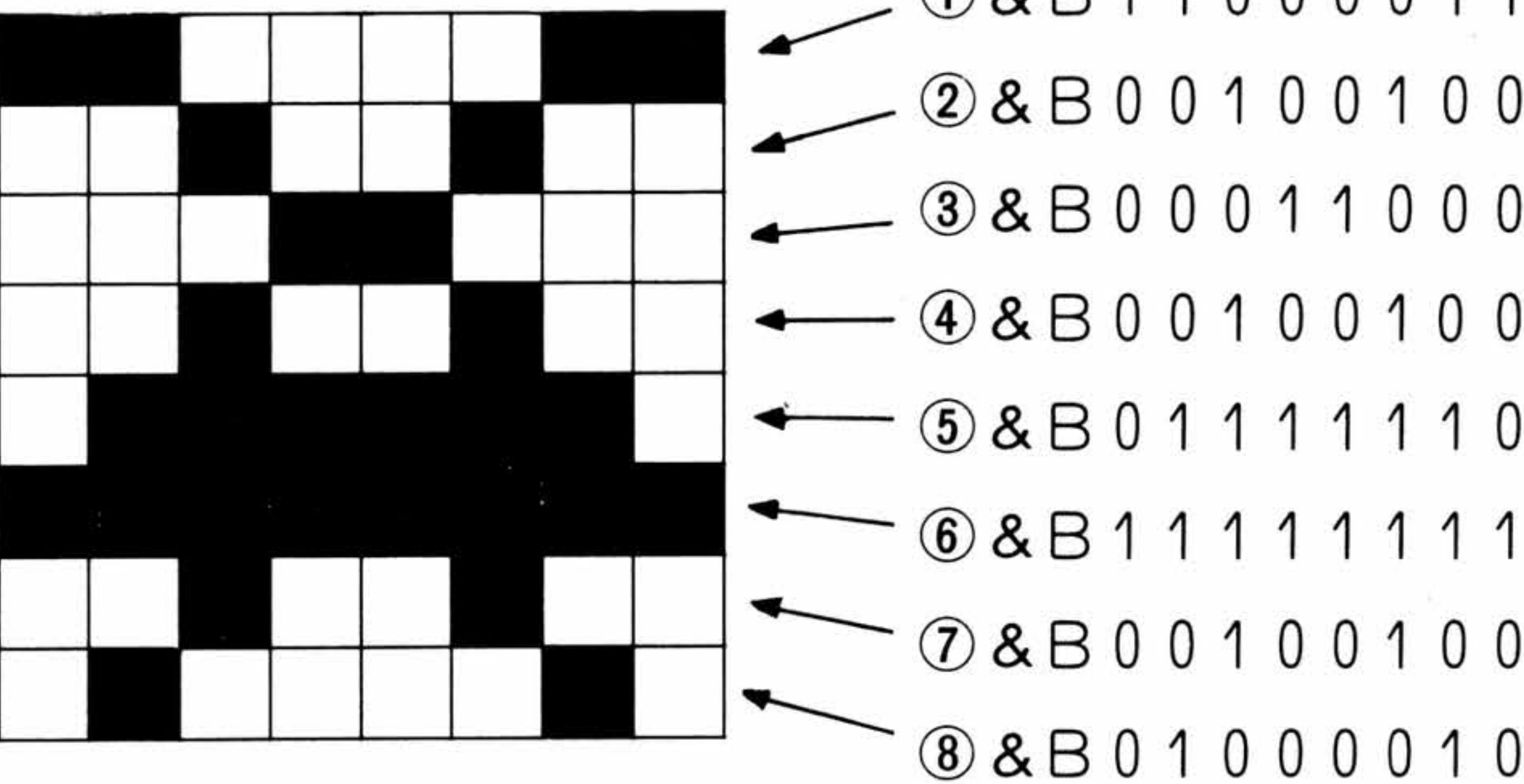
例　SPRITE$（1）=A$+B$

説　明　〈文字式〉で表されるパターンを、〈スプライトパターン番号〉で指定したスプライトに定義します。

〈スプライトパターン番号〉は、SCREEN文で指定したスプライトサイズが0，1のとき0～255の範囲、スプライトサイズが2，3のとき0～63の範囲の数で指定します。

スプライトパターンのデータは8×8の場合8文字分の文字列（8バイト）、16×16ドットの場合32文字分の文字列（32バイト）ですが、データが満たない部分は0で満たされます。データの指定の方法を次の図で説明します。

●8×8ドットのとき

① &B11000011
② &B00100100
③ &B00011000
④ &B00100100
⑤ &B01111110
⑥ &B11111111
⑦ &B00100100
⑧ &B01000010

SPRITE$(n)=①+②+……+⑦+⑧

↑

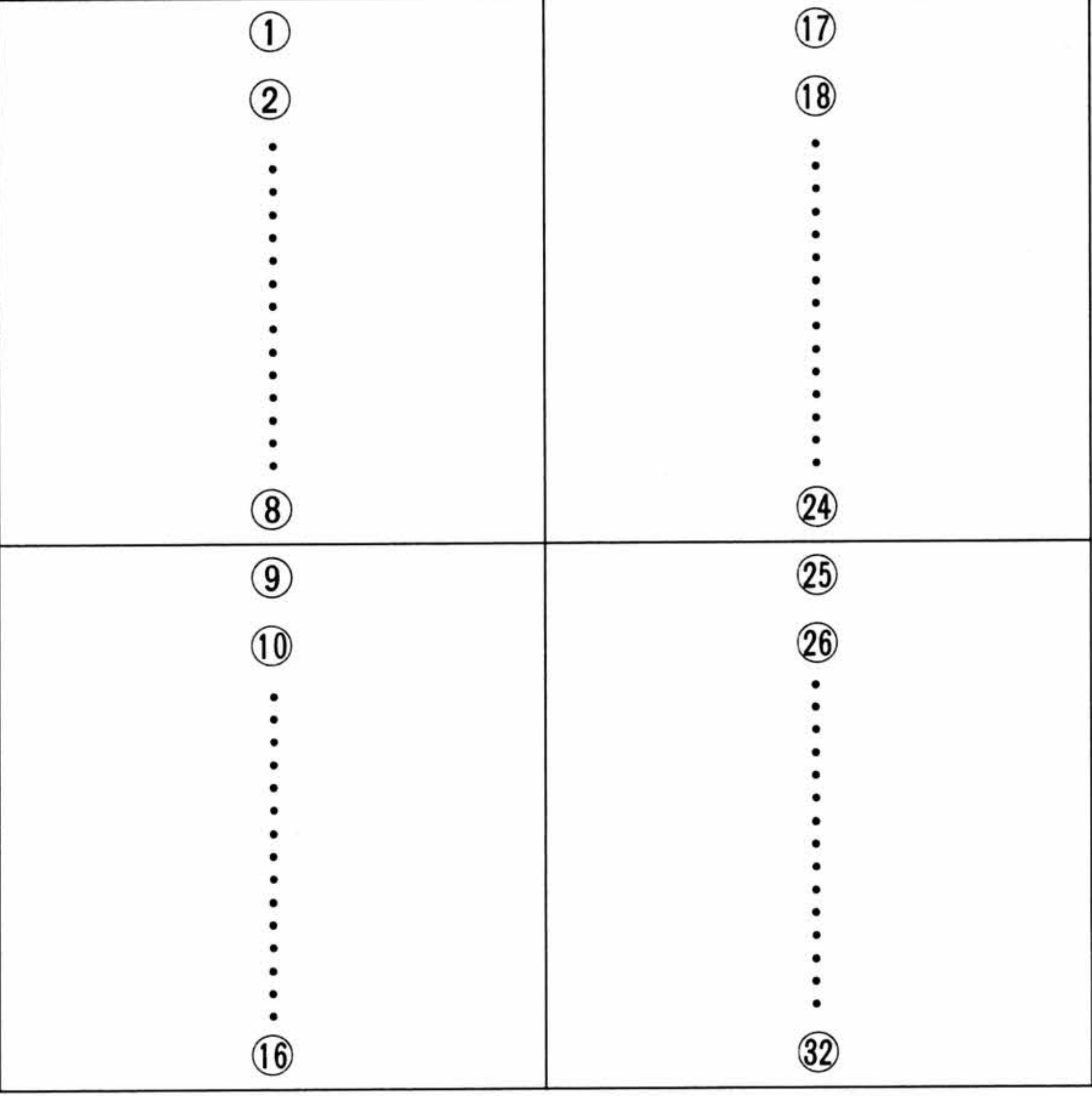
nは0～255

（2進数のほかに、8進数、10進数16進数でも設定できます。）

●16×16ドットのとき

① ② ⋮ ⑧	⑰ ⑱ ⋮ ㉔
⑨ ⑩ ⋮ ⑯	㉕ ㉖ ⋮ ㉜

SPRITE$(n)=①+②+……+㉛+㉜

↑

nは0～63

サンプルプログラム

第２章のサンプルプログラムに使用できる「大きな雲」を描きます。

```
10 '** SPRITE$ **
20 SCREEN 2,3
30 ' ｽﾌﾟﾗｲﾄ ｾｯﾃｲ
40 FOR T=1 TO 4:A$=""
50 FOR K=1 TO 32:READ B$
60 A$=A$+CHR$(VAL("&H"+B$))
70 NEXT K:SPRITE$(T)=A$
80 NEXT T
90 '
100  PUT SPRITE 1,(191, 95),14,1
110  PUT SPRITE 2,(223, 95),14,2
120  PUT SPRITE 3,(191,127),14,3
130  PUT SPRITE 4,(223,127),14,4
140 GOTO 140
150 ' ｸﾓ ﾉ ﾊﾟﾀｰﾝ
160 DATA 00,01,03,07,07,03,07,0F
170 DATA 0F,07,03,01,03,07,07,03
180 DATA E0,F0,FB,FF,FF,FF,FF,FF
190 DATA FF,FF,FF,FF,FF,FF,FF,FF
200 DATA 00,00,80,C0,E0,F0,F0,F8
210 DATA FC,FF,FF,FF,FF,FF,FF,FF
220 DATA 00,00,00,00,00,00,00,00
230 DATA 00,00,80,80,80,C0,C0,F0
240 DATA 03,01,00,01,03,07,0F,0F
250 DATA 07,07,3F,7F,7F,FF,FF,FF
260 DATA FF,FF,FF,FF,FF,FF,FF,FF
270 DATA FF,FF,FF,FF,FF,FF,FF,FF
280 DATA FF,FF,FF,FF,FF,FF,FF,FF
290 DATA FF,FF,FF,FF,FF,FF,FF,FF
300 DATA F8,F8,FC,FC,FC,F8,F0,E0
310 DATA C0,C0,F0,F8,FC,FC,FE,FF
```

このプログラムを第2章のプログラムに追加するときは、
UFO、船、バクハツの各パターンデータに00のデータを
24個追加してください。

SQR

square root（スクエアルート：平方根）　関数

働　き　平方根（スクエアルート）を得ます。

書き方　SQR（〈数式〉）

　例　　PRINT　SQR（3）　　　実行結果……1.7320508075688

説　明　〈数式〉で指定された値の平方根を求めます。

〈数式〉の値は0以上でなければなりません。〈数式〉はどの型の数値でもかまいませんが、結果は倍精度で得られます。また、関数のため、他の命令と組み合せて使用します。

サンプルプログラム

```
10 PRINT "   A   SQR(A)"
20 FOR A=1 TO 15
30 PRINT USING"##.#   ##.###";A;SQR(A)
40 NEXT
```

STICK

stick（スティック：棒・スティック）　関数

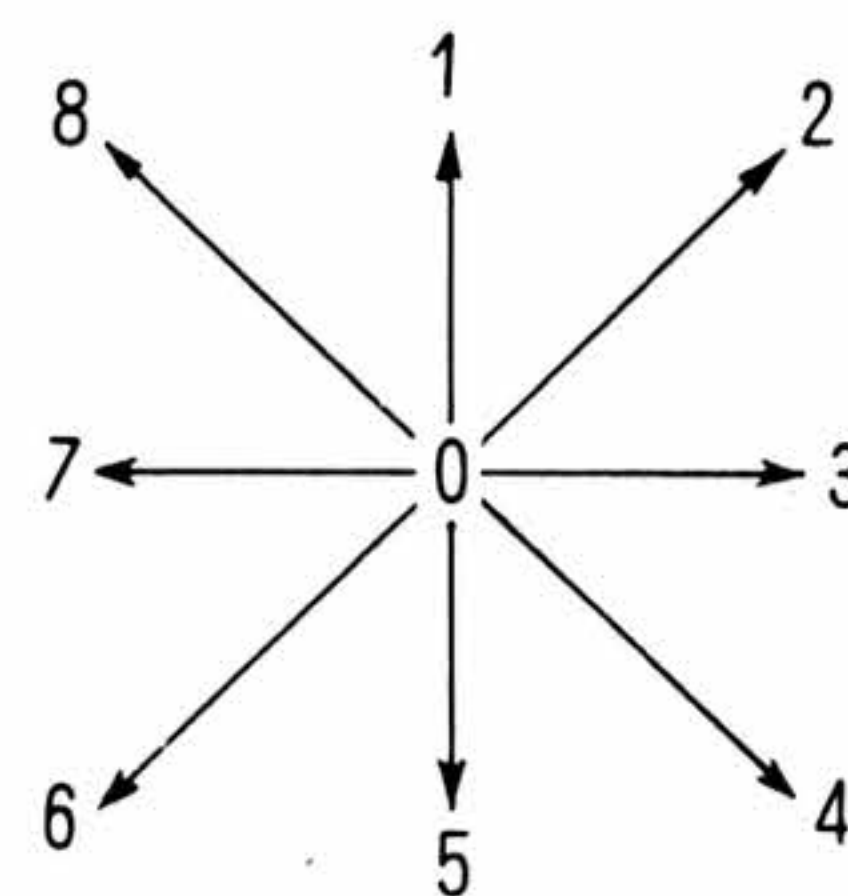
働き　ジョイスティックの押されている方向を調べます。

書き方　STICK（〈ジョイスティック番号〉）

例　A＝STICK（0）

説明　指定したジョイスティックがどの方向へ押されているかを調べ、その結果を数値で得ます。なお、ジョイスティックがどちらへも押されていなければ値は0です。

〈ジョイスティック番号〉は次のように指定します。

0：ジョイスティックの代わりにキーボードのカーソルキーを使用するとき。
1：ジョイスティック1
2：ジョイスティック2

〈方向と値〉

サンプルプログラム　カーソルキーで、ペンの形のスプライトを移動します。

```
10 ' *** STICK(0) ***
20 COLOR 15,1,1:SCREEN 2,2
30 FOR S=1 TO 2:A$=""
40 FOR P=1 TO 32:READ D$
50 A$=A$+CHR$(VAL("&H"+D$)):NEXT
60 SPRITE$(S)=A$:NEXT
70 DATA 1,2,4,D,17,13,21,23,47,4C,F0
80 DATA C0,0,0,0,0,3F,7E,FC,F8,F0,E0
90 DATA C0,80,0,0,0,0,0,0,0,0
100 DATA 0,0,0,0,0,0,0,0,1,2,4,9,13,27
110 DATA 4F,9F,0,0,0,0,10,28,4C,9E,3F
120 DATA 7E,FC,F8,F0,E0,C0,80
130 A=STICK(0)
140 IF A=1 THEN Y=Y-1
150 IF A=2 THEN Y=Y-1:X=X+1
160 IF A=3 THEN X=X+1
170 IF A=4 THEN X=X+1:Y=Y+1
180 IF A=5 THEN Y=Y+1
190 IF A=6 THEN Y=Y+1:X=X-1
200 IF A=7 THEN X=X-1
210 IF A=8 THEN X=X-1:Y=Y-1
220 PUT SPRITE 1,(X,Y+16),15,1
230 PUT SPRITE 2,(X+8, Y),15,2
240 PSET(X-2,Y+29),4
250 GOTO 130
```

S

STOP

stop （ストップ：止まる）　　ステートメント

働　き　プログラムの実行を中断します。

書き方　STOP

説　明　STOP文を実行するとプログラムの実行は中断され、ダイレクトモードとなります。このときは、次のメッセージが表示されます。

Break in xxxxx（xxxxx はストップした行番号）

中断したプログラムは、CONTコマンドによりプログラムの実行を再開することができます。

END文と異なり、STOP文は使用していたファイルを閉じません。

参　照　CONT，END

STOP ON／OFF／STOP

stop on　（ストップ　：ストップキーを許可）
stop off　（ストップオフ：ストップキーを禁止）　ステートメント
stop stop　（ストップストップ：ストップキーを保留）

働　き　CTRL + STOP キー割込みの発生を許可、禁止、または保留します。

書き方　STOP　ON
STOP　OFF
STOP　STOP

説　明　STOP　ON文は CTRL + STOP キー割込みを許可します。この命令を実行しておくと、プログラム実行時に CTRL + STOP キーが押されたときに割込みが発生し、ON　STOP　GOSUB文で定義しておいた割込み処理ルーチンが実行されます。

STOP　OFF文は CTRL + STOP キー割込みを禁止します。

STOP　STOP文は CTRL + STOP キー割込みを保留します。この命令を実行しておくと、以後にかかった CTRL + STOP キー割込みはSTOP　ON文を実行するまで保留され、STOP　ON文を実行するとすぐに割込み処理ルーチンが実行されます。

参　照　ON　STOP　GOSUB，第1章「割込み」

STRIG

striger（スティックトリガ：引き金）　関数

働き ジョイスティックのトリガボタンが押されたかどうかを調べます。

書き方 STRIG（〈ジョイスティックの番号〉）

例　A=STRIG（2）

説明 指定したジョイスティックのボタンあるいはスペースキーが押されたかどうかを調べ、押されていれば－1、押されていなければ0の値が得られます。

〈ジョイスティックの番号〉は次のように指定します。

0：トリガボタンの代わりにキーボードのスペースキーを使う
1：ジョイスティック1のトリガボタン1
2：ジョイスティック2のトリガボタン1
3：ジョイスティック1のトリガボタン2
4：ジョイスティック2のトリガボタン2

ただし、ジョイスティックによってはトリガボタン2がないものもあります。
このときは、1と3，2と4は同じ指定となります。

参照 ON STRIG GOSUB，STRIG ON／OFF／STOP

S

STRIG ON／OFF／STOP

striger on　（スティックトリガオン：トリガ割込みの許可）
striger off　（スティックトリガオフ：トリガ割込みの禁止）　ステートメント
striger stop（スティックトリガストップ：トリガ割込みの保留）

働　き　ジョイスティックのトリガボタンが押されたときの割込みを許可、禁止、または保留します。

書き方　STRIG（〈ジョイスティックの番号〉）ON
STRIG（〈ジョイスティックの番号〉）OFF
STRIG（〈ジョイスティックの番号〉）STOP

例　STRIG（0）ON

説　明　STRIG ON／OFF／STOP文は指定するジョイスティックのトリガボタンあるいはスペースキーが押されたときに発生する割込みを許可、禁止、または保留します。

〈ジョイスティックの番号〉は次のように指定します。

0：トリガボタンの代わりにキーボードのスペースキーを使う
1：ジョイスティック1のトリガボタン1
2：ジョイスティック2のトリガボタン1
3：ジョイスティック1のトリガボタン2
4：ジョイスティック2のトリガボタン2

ただし、ジョイスティックによってはトリガボタン2がないものもあります。

STRIG ON文はトリガボタン割込みを許可します。この命令を実行しておくと、プログラム実行中にジョイスティックのトリガボタンが押されたときに割込みが発生し、ON STRIG GOSUB文で定義しておいた割込み処理ルーチンが実行されます。

STRIG OFF文はトリガボタン割込みを禁止します。

STRIG STOP文はトリガボタン割込みを保留します。この命令を実行しておくと、以後にかかったトリガボタン割込みはSTRIG ON文を実行するまで保留され、STRIG ONを実行するとすぐに割込み処理ルーチンが実行されます。

参　照　ON STRIG GOSUB，第1章「割込み」

STR$

string$（ストリングドル：文字列）　関数

働　き　数値を文字列に変換します。

書き方　STR$（〈数式〉）

例　A$=STR$(123)　　実行結果……文字変数A$は“123”です。

説　明　〈数式〉を10進数で表わされた文字列に変換します。〈数式〉の値は整数型または実数型（倍精度、単精度）のいずれの数値型でも使うことができます。

注意！　STRING$関数と区別してください。

参　照　VAL，STRING$

サンプルプログラム

```
10 '** STR$ **
20 A=12:B=34
30 PRINT A+B
40 PRINT STR$(A)+STR$(B)
50 END
```

STRING$

string$（ストリングドル：文字列）　関数

働　き　1文字を指定する数だけ繰り返した文字列を得ます

書き方　STRING$（〈繰り返しの数〉，｛〈文字式〉｜〈キャラクターコード〉｝）

例　A$＝STRING$（10，“＋”）　　実行結果……A$は「＋＋＋＋＋＋＋＋＋＋」です。

説　明　〈繰り返しの数〉は0～255の範囲で指定します。電源投入時は0～200の範囲で、200以上に変更するにはCLEAR文を使います。
〈文字式〉および〈キャラクターコード〉は文字を指定します。
〈文字式〉の場合は〈文字列〉の最初の1文字が有効です。
〈キャラクターコード〉の場合は値が0から255の範囲に限られ、この値をキャラクタコードとみなし、対応する文字を採用します。

参　照　STR$，CLEAR

サンプルプログラム

```
10 '** STRING$ **
20 FOR F=1 TO 20
30 PRINT USING"##";F;
40 PRINT STRING$(F,"*")
50 NEXT
```

SWAP

swap （スワップ：交換）　ステートメント

働き　2つの変数の値を変換します。

書き方　SWAP 〈変数〉,〈変数〉

例　SWAP A, B

説明　交換する2つの変数の型が一致しているならば、どの型でもSWAP文でその値を交換することができます(整数、単精度、倍精度、文字、および配列変数)。2つの変数の形が一致していないと "Type mismatch" エラーが起こります。
また、SWAP命令を実行する前に変数の設定（=文やLET文など）をしておかなければなりません。

サンプルプログラム　円盤のスプライトが動きます。

```
10 '** SWAP **
20 SCREEN 2,1
30 A$=CHR$(&H24)+CHR$(&H18)+CHR$(&H3C)+C
HR$(&H66)+CHR$(&HDB)+CHR$(&H7E)+CHR$(&H2
4)
40 SPRITE$(1)=A$
50 X1=0:X2=255:K=4
60 FOR X=X1 TO X2 STEP K
70 Y=90-RND(1)*10
80 PUT SPRITE 1,(X,Y),15,1
90 NEXT X
100 SWAP X1,X2:K=-K
110 GOTO 60
```

TAB

tab（タブ）　関数

働き　指定された水平位置まで空白を出力します。

書き方　TAB（〈数式〉）

例　PRINT　TAB（8）；"MSX"

説明　TABはPRINT文やLPRINT文中で使用され、〈数式〉で指定される水平位置まで空白を出力しながらカーソルを移動します。

〈数式〉の値の範囲は0～255までです。

TABは指定した桁から文字列や数値を表示したいときに使用します。

参照　SPC

サンプルプログラム

```
10 '** TAB **
20 PRINT
30 PRINT "##",TAB(8)"##"
40 PRINT "##","##"
50 PRINT "##"TAB(9)"##"
60 PRINT "##";TAB(9);"##"
```

TAN

tangent（タンジェント：正接）　関数

働き　正接（タンジェント）を得ます。

書き方　TAN（〈数式〉）

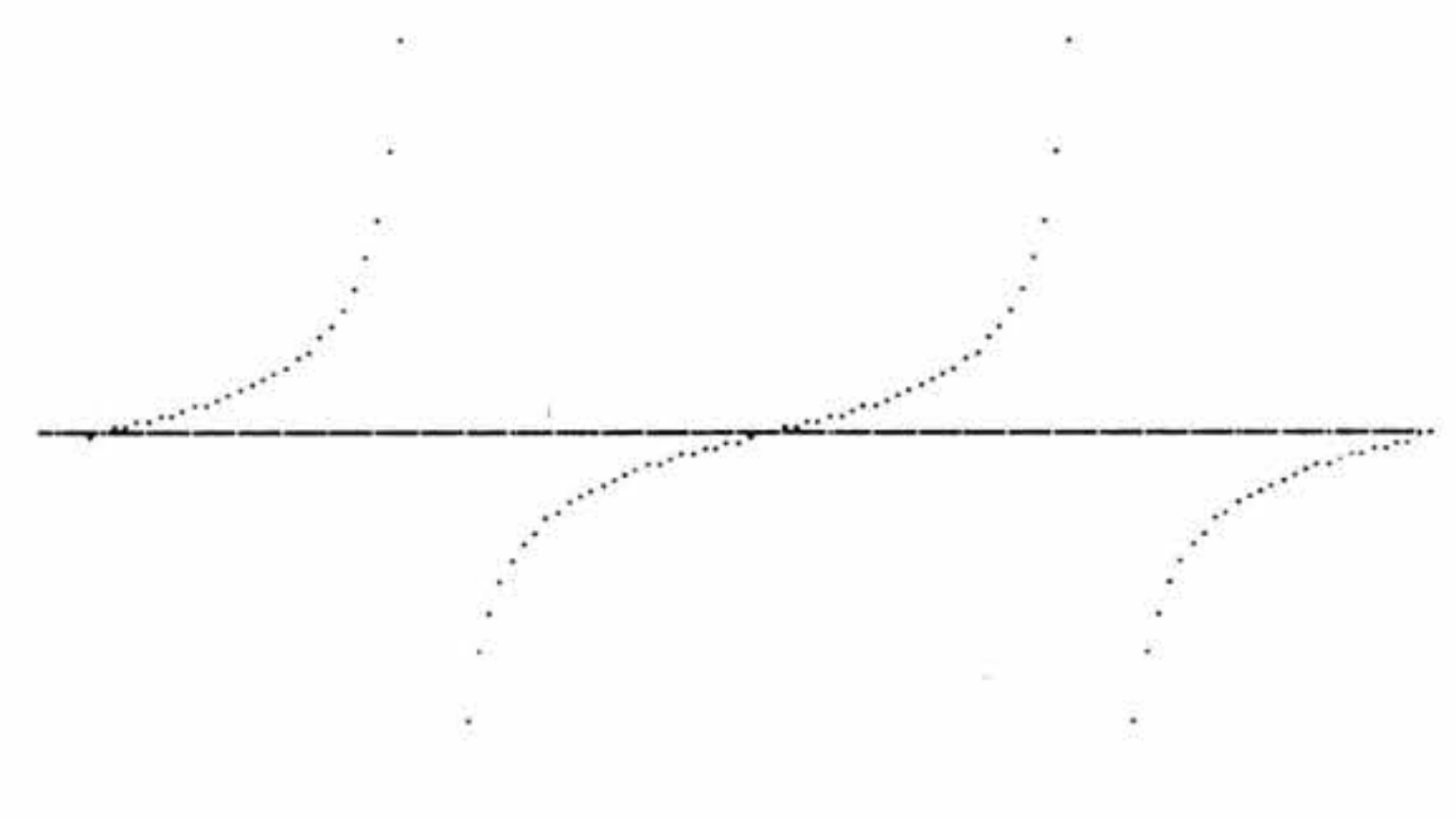
例　A＝TAN（123）

説明　〈数式〉の正接（タンジェント）を求めます。〈数式〉の単位はラジアンです。

〈数式〉は整数型、単精度型、倍精度型のいずれでもかまいませんが、結果は倍精度で得られます。なお、得られる結果の範囲は $-\pi/2$～$\pi/2$ です。

参照　SIN，COS，ATN

サンプルプログラム　正接曲線を描きます。

```
10 '** TAN **
20 SCREEN 2:P=3.1416:X=5
30 LINE(5,100)-(250,100)
40 FOR F=-P TO P STEP P/59
50 Y=100-TAN(F)*10
60 PSET(X,Y)
70 X=X+2
80 NEXT
90 GOTO 90
```

TIME

time（タイム：時間）　　　　　　　　　　　　　　　　　　　　システム変数

働　き　インターバルタイマです。

書き方　TIME〔＝〈数式〉〕

例　PRINT　TIME
TIME＝0

説　明　TIMEは1/60秒ごとに値が1づつカウントアップされるシステム変数です。TIMEが持つ値は0から65535までで、この範囲の値をTIMEに代入することもできます。

プログラムの実行時間を計りたいとき、計りたい処理の一番始めにTIME＝0を代入し、処理の最後でTIMEの内容を調べれば1/60秒単位の時間を得ることができます。

このタイマはVDPが1/60秒ごとに割込みをかけるのをきっかけに値を更新しているため、カセットテープを操作している間など、割込みをかけることができない間はタイマの値は更新されません。

参　照　ON INTERVAL GOSUB，第2章，SET（MSX2のみ）

サンプルプログラム

TRON／TROFF

trace on/off（トレイス オン/オフ：追跡開始/終了）　　　　　　　コマンド

働　き　プログラムの実行状態を追跡する。

書き方　TRON
TROFF

説　明　TRON文を実行すると、以後実行したプログラムの行番号が画面に表示されます。それぞれの行番号は角カッコに囲まれて表示されます。TROFF文またはNEWコマンドを実行するとリセットされます。

TRON文はプログラムのデバッグを助けるために使います。

TRON文の実行はダイレクトモードでもプログラムモードでも可能です。

注意！　行番号はテキストモードの画面に表示されます。グラフィックモードの画面には表示されません。

USR

usr （ユーザー：使用者） 関数

働 き 機械語で作られたサブルーチンを呼び出します。

書き方 USR〔〈番号〉〕（〈引数〉）

例 I=USR 3(J)

説 明 あらかじめメモリ中に準備しておいた機械語サブルーチンを呼び出します。機械語サブルーチンの実行開始番地はDEF USR文で前もって設定されていなければなりません。

〈番号〉は0から9までの値で、DEF USR文により定義された番号に対応しています。〈番号〉が省略された場合には0と解釈されます。

〈引数〉はBASICから機械語サブルーチンへの値の受け渡しをします。なお、引数は省略することができませんので、呼び出すサブルーチンに引数を渡す必要がない場合でもダミーの引数を指定します。

参 照 DEF USR

参 考 USR関数を実行すると、レジスタAに引数の型がセットされ、またレジスタHLには引数のアドレス（文字列の場合にはDEストリングディスクリプタのアドレス）がセットされ、機械語サブルーチンからの参照ができるようになります。

機械語サブルーチンでの処理の結果を引渡す場合は、引数の内容を書き換えます。このとき、呼び出すときに指定した〈引数〉と異る型に変更することはできません。また、文字列の場合は文字列長を変更することはできません。
整数型の数値は2バイトの2進形式で表され、下位バイト、上位バイトの順で格納されます。

単精度実数型の数値は4バイトで表され、先頭1バイトが指数部、その後に3バイトの仮数部が続きます。指数部の左側1ビット目が0ならば仮数部が正、1ならば仮数部が負です。以後7ビットがE＋62～E－64までの指数を表します。また、仮数部は1桁を4ビットで表わす6桁の2進化10進数で表わされています。

倍精度実数型の数値は8バイトで表され、先頭1バイトが指数部、その後に7バイトの仮数部が続きます。

レジスタA
引数の型

引数の型：2＝整数型
3＝文字列型
4＝単精度実数型
8＝倍精度実数型

レジスタHL
引数のアドレス

整数

1	2	3	4
		下位バイト	上位バイト

単精度実数

1	2	3	4
指数部	仮 数 部		

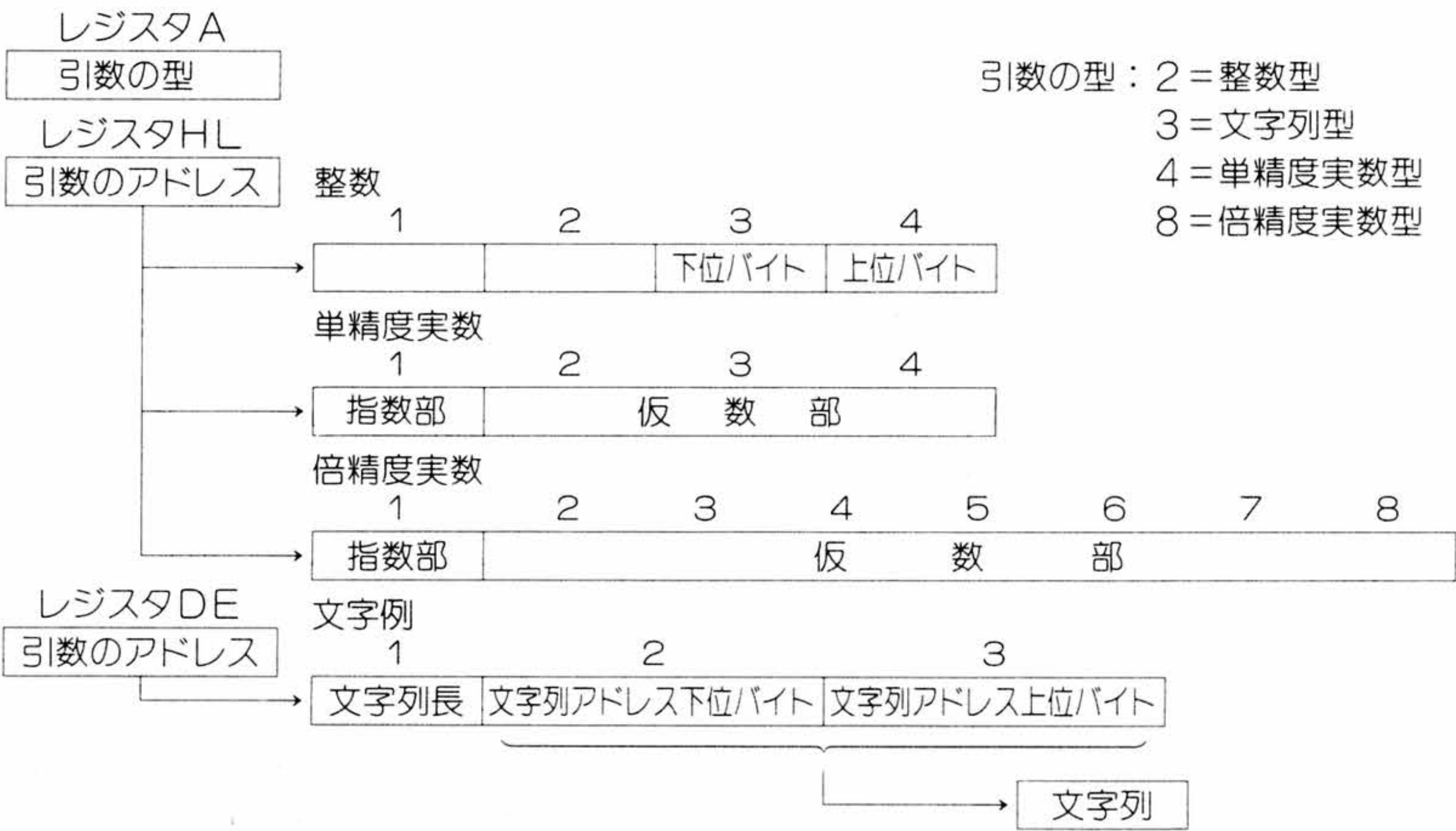
倍精度実数

1	2	3	4	5	6	7	8
指数部	仮 数 部						

レジスタDE
引数のアドレス

文字例

1	2	3
文字列長	文字列アドレス下位バイト	文字列アドレス上位バイト

→ 文字列

VAL

value（バル：数値）　関数

働　き 文字列を数値に変換します。

書き方 VAL（〈文字式〉）

例　A=VAL（"8"）　　実行結果……Aは数値の8です。

説　明 〈文字式〉で表す文字列を数値に変換してその結果を得ます。もし、文字列の最初の文字が＋、－、&、または数字でなければ、VALの値は0になります。ただし、文字列中のスペースは無視されます。

参　考 STR$

サンプルプログラム

```
10 '** VAL **
20 INPUT "16ｼﾝｽｳ・・&H";A$
30 B$="&H"+A$
40 A=VAL(B$)
50 PRINT B$;" ﾊ ";"ｼﾞｭｯｼﾝﾃﾞﾊ";A
```

VARPTR

var pointer（バー ポインター）　関数／Disk　BASIC

働　き 変数の格納されているメモリ番地、ファイルに割り当てられているファイルコントロールブロックの開始番地を求めます。

書き方
1）VARPTR（<変数名>）
2）VARPTR（#<ファイル番号>）

例
1）PRINT　HEX$(VARPTR(X))
2）ADR=VARPTR(#1)

説　明 1）〈変数名〉で指定した変数のデータが格納されている変数領域のメモリ番地を求めます。このとき指定される変数には値が代入されていなければなりません。

←上位アドレス ―――― 下位アドレス→

↓はVARPTRで得られる番地です。

整数型（2バイト）

下位バイト	上位バイト

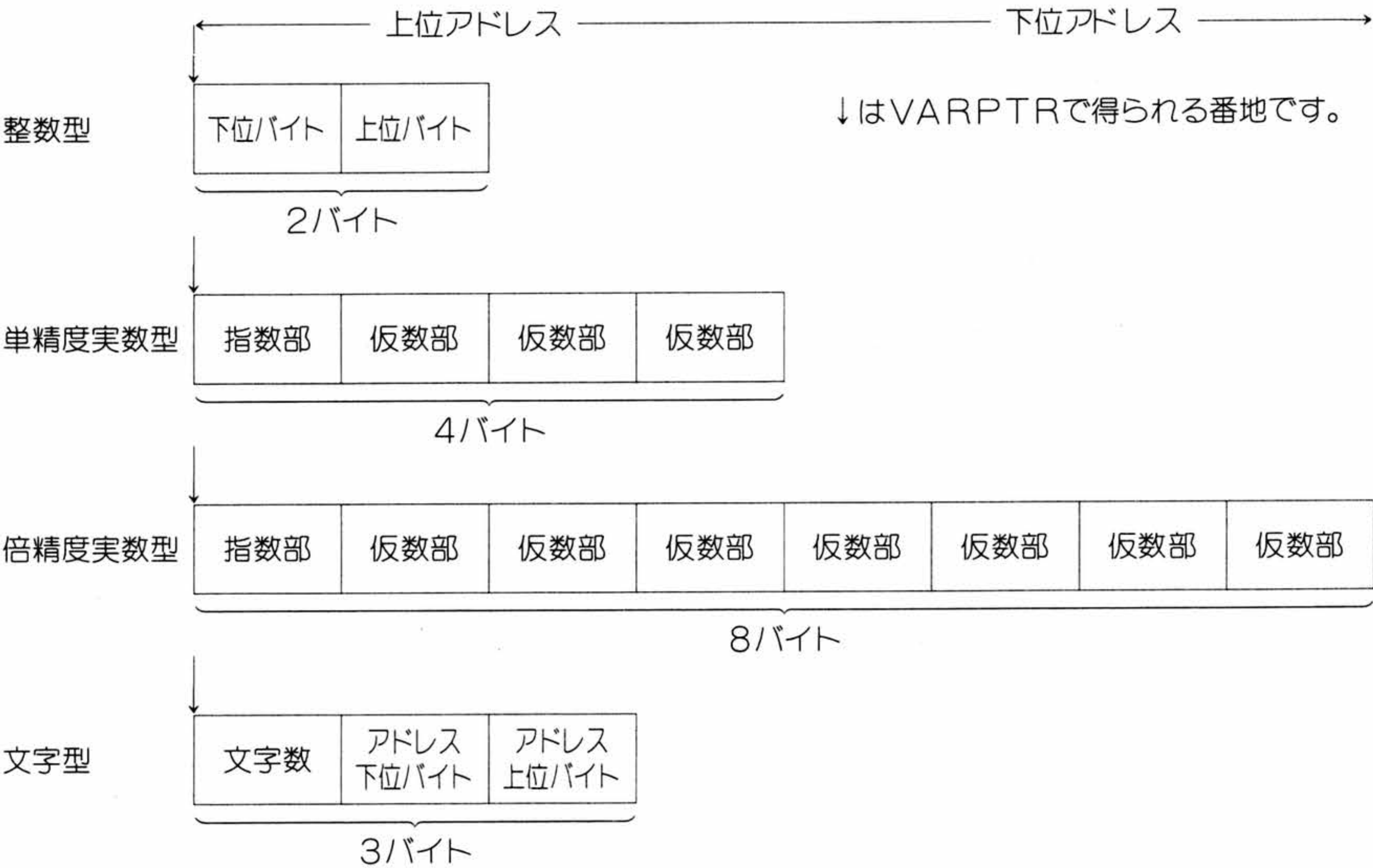
単精度実数型（4バイト）

指数部	仮数部	仮数部	仮数部

倍精度実数型（8バイト）

指数部	仮数部	仮数部	仮数部	仮数部	仮数部	仮数部	仮数部

文字型（3バイト）

文字数	アドレス下位バイト	アドレス上位バイト

2）指定した〈ファイル番号〉に割当てられているファイルコントロールブロックの開始番地を得ます。ファイルコントロールブロックとは、ファイルの入出力のときに使われるバッファのことです。

サンプルプログラム　変数Aと、ファイル番号3の番地を得ます。

```
10 '** VARPTR **
20 MAXFILES=9
30 A=3
40 AD=VARPTR(A)
50 BD=VARPTR(#3)
60 PRINT "A=&H";HEX$(AD)
70 PRINT "B=&H";HEX$(BD)
```

VDP

VDP（ブイ・デー・ピー）　　システム変数

働き VDPのレジスタへデータを直接書き込んだり参照したりします。

書き方 VDP（〈レジスタ番号〉）

例 A＝VDP（5）

説明 VDP（ビデオディスプレイプロセッサ）の機能はBASICで用意しているステートメントや関数でほとんど使うことができるようになっていますが、このシステム変数は、自分でVDPのレジスタを操作するときに用います。

このシステム変数はVDPについての知識を得てから使用してください。

〈レジスタ番号〉は次のように指定します。

0　各種機能、表示モード
1　各種機能、表示モード
2　パターン名称テーブル・ベースアドレス
3　カラーテーブル・ベースアドレス
4　パターンジェネレータ・ベースアドレス
5　スプライト属性テーブル・ベースアドレス
6　スプライトジェネレータ・テーブル
7　テキストカラー、バックドロップカラー
8　割込みフラグ／第5スプライト情報／衝突フラグ（読み出し専用）

MSX2では、−1〜−9，0〜24，および33〜48の〈レジスタ番号〉を指定できます。

VPEEK

video peek（ビデオピーク：ビデオ情報の読み出し）　関数

働　き V（ビデオ）RAM の指定した番地の内容を読み出します。

書き方 VPEEK（<番地>）

例 A%=VPEEK（1）

説　明 画面の操作はBASICで用意しているステートメントや関数ですべて行えるようになっていますが、この関数はVRAM上にある画面データ（名称テーブル、カラーテーブル、パターンジェネレータテーブル、スプライト属性テーブル、スプライトパターンテーブルなど）を直接参照します。

<番地>はVRAMの先頭番地を0とする、0～16383（&HO～&H3FFF）の値を持つ数式で指定します。また、MSX2では0～65535の値でSET PAGEにより、変ります。

読み出すデータは1バイト分（0～255の数値）です。

参　照 VPOKE，BASE，223ページ

VPOKE

video poke（ビデオポーク：ビデオ情報の書き込み）　ステートメント

働　き V（ビデオ）RAMの指定した番地にデータを書き込みます。

書き方 VPOKE〈番地〉、〈データ〉

例 VPOKE　1%、&HFF

説　明 画面の操作はBASICで用意しているステートメントや関数ですべて行えるようになっていますがこの関数はVRAM上にある画面データを直接書き換えることもできます。

〈番地〉はVRAMの先頭番地を0とする0～16383の値を持つ数式で指定します。
また、MSX2では0～65535の値でSET PAGEにより、変ります。

〈データ〉は書き込むデータ1バイト分（0～255）の値を持つ数式です。

参　照 VPEEK，BASE

WAIT

wait（ウエイト：待つ）　ステートメント

働　き コンピュータの指定した入力ポートから特定のデータを読み込む間プログラムの実行を中断します。

書き方 WAIT〈ポート番号〉，〈式1〉〔,〈式2〉〕

例 WAIT　1，&H22，&H22

説　明 WAIT文を実行すると、指定した入力ポートのビットパターンが指定した状態になるまでプログラムの実行が中断されます。

〈ポート番号〉で指定したポートから読み込んだデータと〈式2〉とのXORの結果を得、さらに〈式1〉とのANDが取られます。もしその結果が0（偽）ならBASICはもう一度ポートの状態を読み込み、同じ操作を繰り返します。もし結果が0でなければ（真）、プログラムの実行は次の文にうつります。〈式2〉が省略された場合は0とみなします。

注意！ WAIT文の実行により無限ループにはいってしまう場合があります。この場合にはリセットボタンを押してください。

WIDTH

width（ウィズス：広さ、幅）　ステートメント

働　き 画面に表示する桁数を指定します。

書き方 WIDTH〈桁数〉

例　WIDTH30　　実行結果……1行の文字数を30字とします。

説　明 画面に表示する1行の桁数を設定します。

〈桁数〉の範囲は、40×24テキストモードでは1～40まで、32×24文字のテキストモードでは1～32までです。
文字数の設定は、40×24のテキストモードと32×24のテキストモードとは別々に設定でき40×24のときにWIDTH30としても、32×24のときの設定は変りません。

MSX2では、SCREEN 0で41～80までの値を指定できます。このとき画面は、80×24のモードとなります。

参　照 SCREEN

サンプルプログラム

```
10 SCREEN 0
20 FOR W=39 TO 5 STEP -1
30 WIDTH W
40 FOR T=0 TO 500:NEXT T
50 NEXT W
```

第4章
サンプルプログラム

―― サンプルプログラム ――

この章では、第3章で説明したステートメントや関数を組み合わせたプログラムをいくつか紹介します。サンプルプログラムはそのままでも使うことができますが、改良したり、他のプログラムの一部として組み込むこともできます。

説　明 ショパンの「華麗なる大円舞曲」の8小節をPLAY文を使って演奏します。ここでは，DATA文中に1小節分ごとの音楽データを用意し、READ文で順次読み出して3重和音で演奏します。

```
10 '** PLAY ﾜﾙﾂ **
20 CLS:PRINT"ﾜﾙﾂ"
30 READ A$,B$,C$
40 IF A$="" THEN END
50 PLAY A$,B$,C$
60 GOTO 30
70 '
80 'ﾃﾞｰﾀ
90 '
100 DATA V13,V10,V10
110 DATA O4L4B-O5D8E-8F
120 DATA RRO5L4D
130 DATA RRR
140 DATA O4L4B-O5E-8F8G
150 DATA RRO5L4E-
160 DATA RRR
170 DATA O4L4B-O5F8G8A-
180 DATA RRO5L4F
190 DATA RRR
200 DATA O5L16B-4B-8R8B-R48B-R48
210 DATA O5L16G4G8R8GR48GR48
220 DATA O5L16D-4D-8R8D-R48D-R48
230 DATA O5L4B-O6C8O5B-8A-
240 DATA O5L2GR
250 DATA O5L2D-C4
260 DATA O5L4A-B-8A-8G
270 DATA O5L2C-O4B-4
280 DATA RRR
290 DATA O5L4GA-8G8F
300 DATA O4L2B-A-4
310 DATA RRR
320 DATA O5L4FG8F8E-
330 DATA O4L2A-G4
340 DATA RRR
350 DATA ""
360 DATA ""
370 DATA ""
```

SPRITE機能・COLOR文

説　明 SPRITE機能とCOLOR文を使って、画面の色調整用のプログラムを作ります。

```
10 ' *** COLOR ***
20 COLOR 15,1,1:SCREEN2,2
30 OPEN "GRP:" FOR OUTPUT AS#1
40 FOR S=1 TO 2:A$=""
50 FOR P=1 TO 32:READ D$
60 A$=A$+CHR$(VAL("&H"+D$)):NEXT
70 SPRITE$(S)=A$:NEXT
80 FOR K=15 TO 2 STEP -1:Y=K*11+13
90 FOR X=10 TO 75+K*5 STEP 2
100 PUT SPRITE K,(X,Y),K,1
110 PUT SPRITE K+15,(X+8,Y-16),K,2
120 LINE(X-4,Y+3)-(X-2,Y+12),K,BF:NEXT
130 READ D$:PSET(X+30,Y-13),1:PRINT#1,D$
140 S=50+K*2:PLAY"V9N=S;32":NEXT
150 DRAW"BM31,15":PRINT#1,"ﾄｳﾒｲ"
160 DRAW"BM31,26":PRINT#1,"ﾌﾞﾗｯｸ"
170 DRAW"BM16,0":PRINT#1,"Push RETURN ke
y"
180 IF INKEY$<>CHR$(13) THEN 180
190 COLOR 15,4,7:END
200 DATA 1,2,4,D,17,13,21,23,47,4C,F0
210 DATA C0,0,0,0,0,3F,7E,FC,F8,F0,E0
220 DATA C0,80,0,0,0,0,0,0,0,0
230 DATA 0,0,0,0,0,0,0,0,1,2,4,9,13,27
240 DATA 4F,9F,0,0,0,0,10,28,4C,9E,3F
250 DATA 7E,FC,F8,F0,E0,C0,80
260 DATA ﾎﾜｲﾄ,ｸﾞﾚｰ,ﾏｾﾞﾝﾀ,ﾀﾞｰｸｸﾞﾘｰﾝ
270 DATA ﾗｲﾄｲｴﾛｰ,ﾀﾞｰｸｲｴﾛｰ,ﾗｲﾄﾚｯﾄﾞ
280 DATA ﾚｯﾄﾞ,ｼｱﾝ,ﾀﾞｰｸﾚｯﾄﾞ,ﾗｲﾄﾌﾞﾙｰ
290 DATA ﾀﾞｰｸﾌﾞﾙｰ,ﾗｲﾄｸﾞﾘｰﾝ,ｸﾞﾘｰﾝ
```

SOUND文

説　明 SOUND文では周波数、エンベロープ形状、ノイズ周波数などを細かく指定して音を出しますので、効果音も作れます。
ピアノ、オルガン、トレモロ、パイプオルガンなどが演奏できます。

```
10 ' *** KEYBOARD MUSIC ***
20 CLEAR 4000:DEFINT A-Z:DIM S(9,14,1)
30 ON STOP GOSUB 830:STOP ON
40 FOR S=1 TO 9:FOR K=1 TO 14:READ SS:IF
 SS=0 THEN 60
50 SS=1789773#/(16*SS):S(S,K,0)=SS MOD 2
56:S(S,K,1)=SS¥256
60 NEXT K,S
70 OPEN"GRP:"FOR OUTPUT AS#1:COLOR15,4,4
:SCREEN 2,1,0
80 '
90 FOR X=8 TO 210 STEP 49
100 LINE(X,10)-(X+49,40),15,B:NEXT X
110 DRAW"BM25,15":PRINT #1,"F1     F2
F3     F4     F5"
120 DRAW"BM15,30":PRINT #1,"ﾋﾟｱﾉ   ｵﾙｶﾞﾝ
ﾄﾚﾓﾛ  ﾊﾟｲﾌﾟ ﾒﾄﾛﾉｰﾑ"
130 LINE(8,45)-(75,70),,B:LINE -STEP(40,
-25),,B:LINE-STEP(40,25),,B:LINE-STEP(99
,-25),,B
140 DRAW"BM22,47":PRINT #1,"ﾃﾝﾎﾟ    DOWN
 UP"
150 DRAW"BM13,60":PRINT #1,"(ｶｰｿﾙｷｰ)   <
  >"
160 DRAW"BM160,54":PRINT #1,"tempo= 120"
170 DRAW"BM8,75C15R245D15L190U15D15L55U1
5"
180 DRAW"BM11,79":PRINT #1,"ｵｸﾀｰﾌﾞ   1   2
   3   4   5   6   7   8"
190 '
200 FOR X=35 TO 205 STEP 16
210 LINE(X,105)-(X+14,175),15,BF:NEXT
220 DRAW"BM40,177":PRINT #1,"ﾀ ﾁ ﾂ ﾃ ﾄ ﾔ
 ﾕ ﾖ ﾜ ｦ ﾝ"
230 READ X:IF X=0 GOTO 250
240 LINE(X,105)-(X+9,140),2,BF:GOTO 230
250 DRAW"BM30,95":PRINT #1,"ｻ   ｽ ｾ   ﾏ
ﾐ ﾑ   ﾓ ﾞ"
260 COLOR 1:DRAW"BM56,164":PRINT #1,"●
             ●"
270 '
280 SPRITE$(1)=CHR$(4)+CHR$(6)+CHR$(5)+C
HR$(5 )+CHR$(53 )+CHR$(124)+CHR$(120)+CH
R$(48)
290 SPRITE$(2)=CHR$(32)+CHR$(112)+CHR$(2
16)+CHR$(112)+CHR$(32)
```

4章

SOUND文

```
300 PUT SPRITE 1,(40,10),8,1:PUT SPRITE
3,(137,77),11,2
310 COLOR 15:SOUND 0,0:SOUND 1,0:SOUND 7
,248:SOUND 8,16:SOUND 12,30
320 ON KEY GOSUB 530,560,580,600,620
330 FOR K=1 TO 5:KEY (K) ON:NEXT:ON INTE
RVAL=30 GOSUB 680:INTERVAL OFF
340 MF=0:S=4:R=0:TE=120:B$="サタシチスツセテソトマヤ
ミ1ムヨメワモヲ゛ン":C$="アイウエオナニヌ"
350 '
360 ' ** INKEY CHECK **
370 D$=" "
380 POKE&HFCAB,255:OUT171,12
390 POKE&HFCAC,255:OUT160,15:OUT161,0
400 A$=INKEY$:IF A$=""GOTO 500
410 B=INSTR(B$,A$):C=INSTR(C$,A$)
420 IF A$=CHR$(28) GOTO 650 ELSE IF A$=C
HR$(29) GOTO 660
430 IF B=0 THEN PUT SPRITE 4,(260,200),0
,2 ELSE PUT SPRITE 4,(21+8*B,162-55*(B M
OD 2)),11,2
440 IF 1<=C THEN  S=C:PUT SPRITE 3,(41+2
4*C,77),11,2
450 IF B>14 THEN FF=1:B=B-14 ELSE FF=0
460 IF D$=A$ GOTO 480 ELSE SOUND 13,13
470 SOUND 0,S(S+FF,B,0):SOUND 1,S(S+FF,B
,1):SOUND 13,R
480 D$=A$:GOTO 380
490 '
500 FOR I=&HFBE5 TO &HFBEA:X=PEEK(I):IF
X<>255 THEN 380
510 NEXT
520 SOUND 0,0:SOUND 1,0:GOTO 370
530 ' ** PIANO **
540 SOUND 8,16:SOUND 12,30:SOUND 0,0
550 SOUND 1,0:R=0:PUT SPRITE 1,(40,10),8
:RETURN
560 ' ** ORGAN **
570 SOUND 8,12:SOUND 0,0:SOUND 1,0:R=9:P
UT SPRITE 1,(89,10),8:RETURN
580 ' ** TREMOLO **
590 SOUND 8,16:SOUND 13,0:SOUND 12,1:SOU
ND 0,0:SOUND 1,0:R=10:PUT SPRITE 1,(138,
10),8:RETURN
600 ' ** PIPE ORGAN  **
610 SOUND 8,16:SOUND 13,0:SOUND 12,10:SO
UND 0,0:SOUND 1,0:R=13:PUT SPRITE 1,(187
,10),8:RETURN
620 ' ** TEMPO **
630 IF MF=0 THEN MF=1:INTERVAL ON:PUT SP
RITE 2,(235,10),11,1:RETURN
```

```
640 IF MF=1 THEN MF=0:INTERVAL OFF:PUT S
PRITE 2,(235,10),0:RETURN
650 TE=TE+4:IF TE>256 THEN TE=256
660 TE=TE-2:IF TE<34 THEN TE=34
670 LINE(210,50)-(240,65),4,BF:DRAW"BM20
8,54":PRINT #1,TE:ON INTERVAL=(3600/TE)
GOSUB 680:GOTO 360
680 ' ** METRONOME **
690 SOUND 10,14:PUT SPRITE 2,(235,10),CC
,1:CC=ABS(CC-11):SOUND 10,0:RETURN
700 ' ** SOUND DATA **
710 DATA 29,31,0,32.7,34,36.7,39,41,0,43
,46,49,52,55
720 DATA 58,62,0,64,69,74,78,82,0,87,93,
98,104,110
730 DATA 116,123,0,131,138,146,155,165,0
,174,185,196,208,220
740 DATA 233,247,0,262,277,292,311,330,0
,349,370,393,415,440
750 DATA 466,494,0,523,554,587,622,659,0
,698,740,785,831,880
760 DATA 933,988,0,1046,1108,1174,1244,1
318,0,1398,1480,1568,1662,1760
770 DATA 1866,1977,0,2092,2216,2348,2488
,2636,0,2794,2960,3136,3324,3520
780 DATA 3733,3955,0,4184,4432,4697,4976
,5274,0,5588,5920,6272,6648,7040
790 DATA 7466,7908,0,8368,8864,9395,9952
,10548,0,11176,11840,12544,13296,14080
800 ' ** LINE DATA **
810 DATA 29,61,77,109,125,141,173,189,0
820 ' ** ｶﾅﾓｼﾞ ｶｲｼﾞｮ **
830 POKE&HFCAB,0:OUT171,13
840 POKE&HFCAC,0:OUT160,15:OUT161,128
850 COLOR 15,4,7:END
```

(注)　380, 390行を実行することで、カナ文字の入力状態となりますが、830, 840行を実行すると解除されます。

入力チェックプログラム

説　明 各種のキー入力をチェックするプログラム例です。

INKEY$によるメニュー選択

```
10 CLS
20 LOCATE 5,0:PRINT "* ｼｺﾞﾄｦ ｴﾗﾝﾃﾞ ｸﾀﾞｻｲ
 *"
30 LOCATE 7,3:PRINT "1. INPUTｼﾏｽ"
40 LOCATE 7,5:PRINT "2. DISPLAY"
50 LOCATE 7,7:PRINT "3. PRINTOUT  "
60 LOCATE 7,9:PRINT "4. ｵﾜﾘ"
70 ME=VAL(INKEY$):IF ME=0 THEN 70
80 ON ME GOSUB 130,150,170,100
90 GOTO 20
100 PRINT "* ｼｺﾞﾄﾊ ｵﾜﾘﾏｼﾀ *"
110 END
120 '* INPUT
130 PRINT "INPUT   ｻﾌﾞﾙｰﾁﾝ":RETURN
140 '* DISPLAY
150 PRINT "DISPLAY ｻﾌﾞﾙｰﾁﾝ":RETURN
160 '* PRINT
170 PRINT "PRINT   ｻﾌﾞﾙｰﾁﾝ":RETURN
```

ファンクションキーによるメニュー選択

```
10 ' ** MODE SELECT **
20 PRINT"ｼｺﾞﾄ ｦ ｴﾗﾝﾃﾞ ｸﾀﾞｻｲ｡":KEY OFF
30 KEY 1,"INPUT":KEY 2,"DISPLAY"
40 KEY 3,"PRINT":KEY 4,"LIST"
50 KEY 5,"END"
60 ON KEY GOSUB 90,110,130,150,170
70 FOR K=1 TO 5:KEY(K) ON:NEXT K
80 KEY ON:GOTO 60
90 '* INPUT
100 PRINT "INPUT ｻﾌﾞﾙｰﾁﾝ":RETURN
110 '* DISPLAY
120 PRINT "DISPLAY ｻﾌﾞﾙｰﾁﾝ":RETURN
130 '* PRINT
140 PRINT "PRINT ｻﾌﾞﾙｰﾁﾝ":RETURN
150 '* PRINT
160 PRINT "LIST  ｻﾌﾞﾙｰﾁﾝ":RETURN
170 '* PRINT
180 PRINT "END   ｻﾌﾞﾙｰﾁﾝ":RETURN
```

COLOR

説　明 MSX2の特長であるカラーパレット機能により、おもしろい絵が描けます。

```
5 'color
10 COLOR 15,0,0:SCREEN 5
20 FOR K=0 TO 65 STEP 8
30 FOR C=1 TO 15
40 CIRCLE(127,96),K*2+C,C
50 NEXT C,K
60 FOR C=1 TO 15
70 COLOR=(C,0,0,0)
80 NEXT C
90 FOR C=1 TO 12 STEP 3
100 COLOR=(C,7,0,0)
110 COLOR=(C+1,0,7,0)
120 COLOR=(C+3,0,0,7)
130 FOR K=0 TO 10:NEXT K
140 COLOR=(C,0,0,0)
150 COLOR=(C+1,0,0,0)
160 COLOR=(C+3,0,0,0)
170 NEXT C
180 GOTO 90
```

4章

第5章 資 料

(MSX BASICの概要, 資 料)

本書は、MSX BASICを使ってプログラムを作るときに必要となる基本的な知識と、キャラクターコード表などの資料をまとめています。
第1章、第2章での説明を理解するうえでの参考としてください。

5.1 MSX BASICの概要

5.1.1 コマンドとステートメント

BASICの命令は、コマンドかステートメントあるいは代入を行なう式（＝）で構成されます。コマンドとステートメントは明確に分かれているわけではありませんが、ここでは次のように分類します。

●コマンド

コマンドはプログラムの作成、実行、修正および保存を行ないます。
一般に、ダイレクトモードで使用します。

●ステートメント

１つのステートメントの機能は単純なのでプログラム中に使用し、パソコンにしてもらいたい仕事の命令を構成します。
一般に、プログラムモードで実行しますが、簡単な計算やプログラムの動作をチェックするためにダイレクトモードで実行することもできます。

5.1.2 行

- 行は、パソコンに命令を入力するときの単位で、キー入力を開始してからリターンキーを押すまでを指します。
- １行は最大255文字以内で、１つ以上のコマンド、ステートメントを含むことができます。１行に複数の文を含む場合を複文（マルチステートメント）と呼び、この場合には命令と命令の間をコロン（：）で区切ります。

5.1.3 行番号

- 行を複数個集めてプログラムを構成するとき、行の先頭に命令の実行順序を示す番号を書きます。これを行番号といい、0～65529の整数です。
- 行番号はGOTOなどの分岐命令で指定する分岐先や、LISTなどプログラムの編集時にも使用します。
- LIST，AUTO，DELETEなどの命令で、行番号の代りにピリオド（．）を使用することもできます。ピリオドはエラー発生や［CTRL］＋［STOP］などで、BASICが現在着目している行番号を表します。

　例　LIST.

5.1.4 動作モード

BASIC（Disk-BASIC）で“Ok”という表示がされている状態のとき、行の命令を実行させたりプログラムの作成ができます。
行の命令を実行させる方法として、２つの動作モードがあります。

●ダイレクトモード

行番号をつけずにBASICの文法に合った行を入力すると、リターンキーを押した後、すぐに実行されます。

5-1

●プログラムモード

行番号（0～65529）を行の先頭につけて入力すると、その行はプログラムを構成する行としてメモリに格納されます。格納されたプログラムは、RUN、GOTOなどの命令によって実行されます。

5.1.5 使用できる文字

英大文字、英小文字、ひらがな、カタカナ、数字（0～9）、英記号、かな記号、グラフィック文字が使用できます。
英記号の中で演算子（計算の記号）と次の記号を特殊記号といいます。
なお、演算子については、5.1.10で説明します。

●特殊記号

.（ピリオド）　現在BASICが着目している行番号を表します。LIST、AUTOなどで行番号の代りに使うことができます。

例　LIST.

－（マイナス）　数値・文字の範囲を指定するときに使います。

例　LIST 10－200 ……10行から200行までをリストアウトする
DEFINT A－Z……A－Zの変数を整数型と宣言する

：（コロン）　マルチステートメント中、命令の区切りとして使います。

例　A＝B＊C：GOTO 100

，（カンマ）　INPUT、DATA文などで、データを複数個指定するときの区切りとして使います。

例　INPUT A，B，C

；（セミコロン）　PRINT文などで、数や文字列の区切りとして使います。

例　PRINT A；B＄

'（アポストロフィ）　REM文（注釈文）の代りに使います。

例　'コメントブン

？（疑問符）　PRINT文の代りに使います。

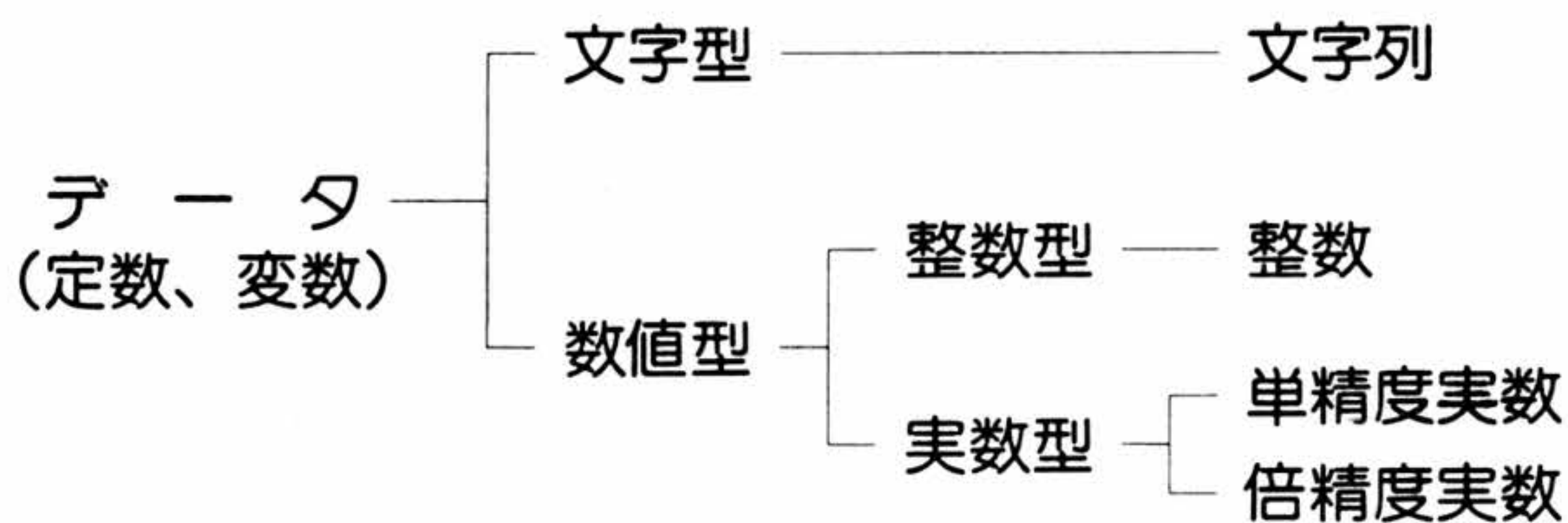
例　？ A＊B

（空　白）　プログラムを見やすくするために、文中に空白を入れることができます。ただし、コマンドなどの予約語中には空白を入れることはできません。また、文字列中の空白は文字としての意味を持ちます。
文中の空白は、命令の実行時には無視されますが、プログラムの格納時に空白の分もメモリが使用されます。

（ヌルストリング）　パソコンで表示できる文字の中で、「なにもない文字」に相当します。プログラム中で文字変数を使うとき、代入を行なう前はこの状態で、変数の中はなにもありません。

5.1.6　データについて

BASICで扱う定数、変数をデータといい、次のように分類することができます。

- **データ（定数、変数）**
  - **文字型** ―― **文字列**
  - **数値型**
    - **整数型** ―― **整数**
    - **実数型**
      - **単精度実数**
      - **倍精度実数**

5.1.7　定数

●文字定数

最大255文字までの文字列で、引用符（”）で囲んで表します。（文字列の中に引用符を入れたいときにはCHR$関数を用います。）
数値を引用符で囲むと文字列とみなし、算術演算はできません。

例　"ABCD"
　　"1234"

●整数型

整数型定数は、10進、2進、8進、16進形式で表すことができます。

i）10進形式

－32768～32767の範囲の整数です。
－32768～＋32767の実数の後に％をつけた数（小数点以下は切り捨てられます）。

例　321
　　3.1416%……3とみなされます

ii）2進形式

先頭に＆Bをつけた0と1の並びで表される2進数です。
＆B0～＆B1111111111111111の範囲ですが、＆B111111111111111以上は負の数です。

例　&B1110……10進数の14です

iii）8進形式

先頭に＆O（オー）をつけた0～7の数字の並びで表される8進数です。
＆O0～＆O177777の範囲です。

例　&O767……10進数では503です。

5-1

iv) 16進形式

先頭に&Hをつけた0～9の数字とA、B、C、D、E、Fの英文字で表される16進数です。
&H0～&HFFFFの範囲ですが、&H8000以上は負の数です。

例　&H 10A ……10進数では266です。
　　&H FFFF……10進数では－1です

注意！ 2進、8進、16進形式で入力された数値でも、PRINT文などで出力すると10進形式で表示されます。

● **単精度実数型**

単精度実数型の定数は、有効桁数6桁の数です。次のいずれかで表します。
i) 数値の後に感嘆符(!)をつけた実数。数値が6桁を越えていると7桁目を四捨五入します。
ii) 6桁以内の実数とEを使った指数表示の組み合せ。Eの範囲はE－64～E＋62です。

例　3.1416!
　　－25.6E＋32

● **倍精度実数型**

倍精度実数型の定数は、有効桁数14桁の数です。次のいずれかで表します。
i) 7桁以上の実数。数値が14桁を越えていると15桁目を四捨五入します。
ii) 数値の後にシャープ(#)をつけた実数
iii) 7桁以上の実数とEを使った指数表示の組み合せ。Eの範囲はE－64～E＋62です。
Eの代りにDを使用することもできます。

例　3.1415926535
　　1.23456789D＋23

注意！ 特に指定がない場合、実数型の定数・変数は倍精度で記憶・計算されます。

定数	型	形　式	例
数値型定数	整数型	10進形式	123, －456%
		2進形式	&B010
		8進形式	&O177
		16進形式	&HF37F
	実数型	単精度実数型	6.25!, 1.4142E＋10
		倍精度実数型	3.14159#, 2.71828D－25
文字型定数	文字型	文字列	"ABCD", "21595"

5.1.8　変数

変数は、プログラム中で使う文字列や数値をしまっておくことができる入れ物（メモリ上のエリア）で、英数字からなる名前（変数名）によって値の代入・読み出しを行ないます。

変数の値は、代入を行なう前（リセット時、プログラムの実行開始時）には0またはヌルストリングとなります。

●変数名

i）変数名の長さは255文字まで可能ですが、3文字目以降の文字は判別されません。
ii）変数名には英文字、数字を使用しますが、先頭の1文字目は必ず英文字とします。
iii）変数名は予約語（BASICであらかじめ設定されている命令）と同じであったり、予約語を含んでいてはいけません。
iv）変数名を小文字で入力しても、大文字に変換されます。

例　ABC ……変数として使える
　　ABSC……変数として使えない

●変数の型宣言

変数にも、格納するデータの型に応じた型があります。変数の型を宣言するには次の方法があります。

i）型宣言文字（$，%，!，#）をつける。
変数名の後に型宣言文字をつけることで変数の型を区別します。

$（ドル）……………文字型	例　A$	
%（パーセント）……整数型	例　A%	
!（感嘆符）…………単精度実数型	例　A!	
#（シャープ）………倍精度実数型	例　A#	※すべて別の変数を指します

型宣言文字をつけていない変数名は、「#」をつけているのと同じ結果になります。

ii）型宣言文を実行する
プログラム中で、型宣言文を実行することで変数の型を区別します。

DEFSTR（デファインストリング）…………文字型
DEFINT（デファインインテジャー）………整数型
DEFSNG（デファインシングル）……………単精度実数型　※詳しくは、第3章をご覧ください。
DEFDBL（デファインダブル）………………倍精度実数型

●配列変数

配列変数は、いくつかの要素で構成される変数です。配列変数の次元の数はDIM文によってあらかじめ宣言します。配列変数の値を代入したり参照するときには添字を使って何番目の要素なのかを指定します。
3次元までの配列変数については、DIM文を省略して配列変数を使おうとすると自動的に各次元の添字を10として設定します。（詳しくは、第1章、第3章をご覧ください）

●変数と使用メモリ

変数に値を代入したとき、あるいはDIM文を実行したときに、各変数の領域がメモリに確保されます。

i）変数
整数型　　5バイト

単精度実数型	7バイト
倍精度実数型	11バイト
文字型	6+(文字列の文字数)バイト

ii)配列変数

整数型	5+2×(要素数)+2×(次元数)+1バイト
単精度実数型	5+4×(要素数)+2×(次元数)+1バイト
倍精度実数型	5+8×(要素数)+2×(次元数)+1バイト
文字型	5+3×(要素数)+2×(次元数)+1+(各要素に含まれる文字列の総文字数)バイト

5.1.9 型変換

ある型の数値(定数または変数)を、他の型に変換することができます。ただし、数値定数(変数)を文字変数に、またはその逆に変換するときは、STR$、VAL関数を使います。

●数値データを異なる型の変数に代入する場合、数値は代入先の変数の型に変換されます。

例　A%=3.14……A%は3です。

●精度の異なる数値を使った演算では、精度の高い方の型に変換されて演算が行なわれます。ただし、返される数値は、指定のある場合には指定される精度になります。

例

```
10 A!=1/3
20 PRINT A!
RUN
 333333
```

計算は倍精度で行なわれますが、結果は単精度で返されます。

●論理演算の数値は整数に変換され、結果は整数で返されます。

●実数を整数に変換すると小数点以下は切り捨てられます。このとき、結果が－32768～32767の範囲にないとエラーとなります。

●倍精度実数を単精度実数型変数に変換すると、変数の値は有効数字6桁に丸めたものとなります。(7桁目を四捨五入します)

●単精度実数を倍精度実数型変数に変換すると、上位6桁のみが有効となります。

5.1.10 式と演算

定数や変数、文字の組み合せを、演算子を使って1つの値にするものを式といいます。
演算には、算術演算、関係演算、論理演算、関数、文字列の演算があります。

●算術演算

i)つぎの算術演算子を用います。演算の実行順序を変更するときは、カッコ(　)を使用します。カッコの中の演算は他の演算より先に実行されます。

	演算子	演算	例
実	^	指数演算	X^Y
行	−	負号	−X
	*, /	剰算、実数の除算	X*Y, X/Y
順↓	+, −	加算、減算	X+Y, X−Y

ii) 整数の除算と剰余の計算は、¥とMODによって行なわれます。

演算子	演算	例	
¥	整数除算	10 ¥ 3	……10/3=3 余り1で3が得られる。
MOD	整数剰余	10 MOD 3	……10/3=3 余り1で1が得られる。

iii) 0で除算した場合、負のべき剰を行った場合は、"Division by zero" エラーとなります。

iv) 代入や演算の結果が、その変数の型の範囲を越えたとき桁あふれ "Overflow" エラーとなります。

● **論理演算**

論理演算子は複数の条件を調べたり、ビット操作やブール演算を行ったりするのに用います。論理演算の結果は、ビットごとに0または1で得られます。
各論理演算の内容を次に示します。

NOT=not（否定）

X	NOT X
1	0
0	1

XOR=exclusive or（排他的論理和）

X	Y	X XOR Y
1	1	0
1	0	1
0	1	1
0	0	0

AND=and（論理積）

X	Y	X AND Y
1	1	1
1	0	0
0	1	0
0	0	0

IMP =implication（包含）

X	Y	X IMP Y
1	1	1
1	0	0
0	1	1
0	0	1

OR =or（論理和）

X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0

EQV=equivalence（同値）

X	Y	X EQV Y
1	1	1
1	0	0
0	1	0
0	0	1

条件判断文（IF文）では論理演算子を使って、複数の条件を判断することができます。

例）　IF X<0 OR 99<X THEN 1000
　　　　Xが負または、99より大きければ行番号1000へジャンプする。
　　　IF 0<X AND X<100 THEN X=0
　　　　Xが正でかつ、100より小さければXに0を代入する。

5-1

IF NOT (A=0) THEN 20

Aが0でなければ行番号20へジャンプする。

●関数

関数とは、与えられた引数に対して、ある決まった演算を行うもので、この演算の結果を値として得ます。
BASICでは“組込み関数”としてSIN(正弦)、SQR(平方根)などの数値関数やCHR$, LEFT$などの文字列関数が用意されています。
また、BASICは“ユーザー定義関数”としてユーザーが自由に定義できる関数機能ももっています。
数値関数のうち引数を実数で取り得るもの(SIN関数など)は、引数を整数や単精度実数で指定することはできますが、演算は倍精度に変換して行われ、結果は倍精度で得られます。
一般に引数に整数しか取り得ないものは、小数部分を切り捨てて整数に丸めてから演算が行われます。

5.1.11 文字列の演算

●文字列の連結

文字列は演算子(+)を使って連結することができます。

例)

```
10 A$="モジレツ":B$="ヲ":C$="レンケツシマス"
20 D$=A$+" "+B$+" "+C$
30 PRINT D$
RUN
モジレツ ヲ レンケツシマス
```

●文字列の比較

文字列も数値の比較に用いられるものと全く同じ関係演算子を用いて比較することができます。

=, <, >, <>, ><, <=, =<, >=, =>

文字列の場合、それぞれの文字列の最初から1文字ずつ文字の比較を行います。もし相互に全く同じ文字列の場合は、その2つの文字列は等しくなりますが、1個所でも違った場合は、その文字のキャラクタコード(資料のキャラクタコード表参照)の大きい方の文字列が大きくなります。文字列の片方が短くて比較が途中で終わった場合は、短い文字列の方が小さくなります。
文字列の比較においては空白なども意味をもちます。

例)

```
    "AA" < "AB"
"BASIC" = "BASIC"
    "X&" > "X#"
"PEN " > "PEN"
    "cm" > "CM"
 "DESK" < "DESKS"
```

5.1.12 エラーメッセージ

プログラムの実行を中断しなければならないようなエラーが実行時に発生した時、エラーメッセージが画面に表示され、ダイレクトモードに戻ります。
ダイレクトモードでのエラーメッセージの書式は、

XX…………

プログラムモードでの書式は、

XX………… in yyyyy

(XX………はエラーメッセージで、yyyyyはエラーが検出された行番号です。エラーメッセージの内容については、資料の「エラーメッセージ表」を参照してください。)

5.1.13 画面モード

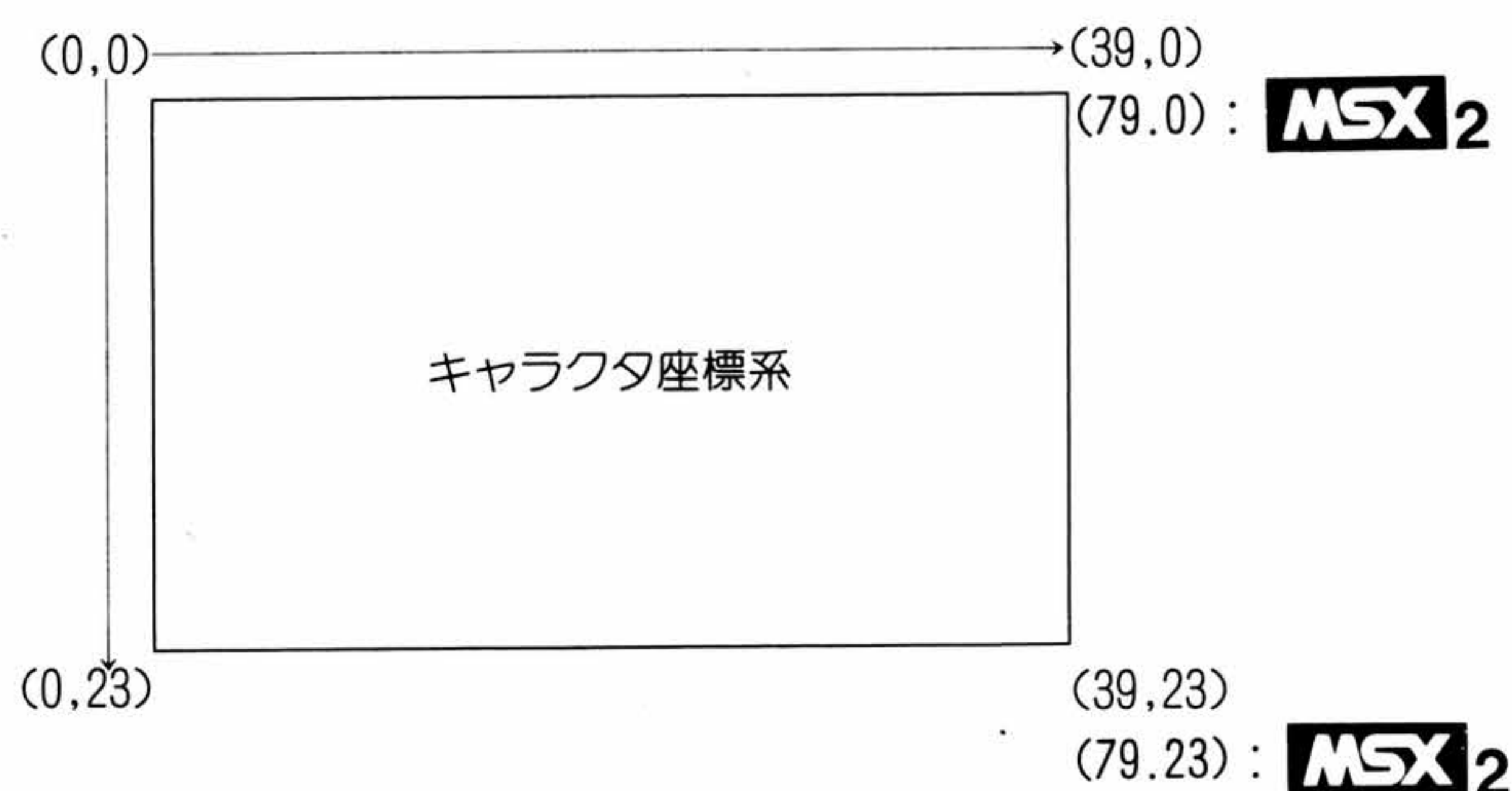
MSX BASICでは次の画面モードがありSCREEN文を使って選択することができます。なおシステム起動時にはテキストモード の状態になっています。

モード	解像度	文字サイズ	表示文字数	スプライト	備考
テキストモード (SCREEN 0)	―――	6×8ドット	最大80×24文字	不可	MSXでは、最大40×24文字
テキストモード (SCREEN 1)	―――	8×8ドット	最大32×24文字	可	―――
高解像度グラフィックモード(SCREEN 2)	256×192ドット	―――	―――	可	8ドットを1単位とする領域ごとに2色まで使うことができる
マルチカラーモード (SCREEN 3)	64×48ブロック	―――	―――	可	ブロックごとに色を指定できる
高解像度グラフィックモード (SCREEN 4)	256×192ドット	―――	―――	可	8ドットを1単位とする領域ごとに2色まで使うことができる(SPRITE機能拡張)
ビットマップグラフィックモード (SCREEN 5)	256×212ドット	―――	―――	可	1点ごとに16色の色の中から自由に使うことができる
ビットマップグラフィックモード (SCREEN 6)	512×212ドット	―――	―――	可	1点ごとに4色の色の中から自由に使うことができる
ビットマップグラフィックモード (SCREEN 7)	512×212ドット	―――	―――	可	1点ごとに16色の色の中から自由に使うことができる
ビットマップグラフィックモード (SCREEN 8)	256×212ドット	―――	―――	可	1点ごとに256色の中から自由に使うことができる

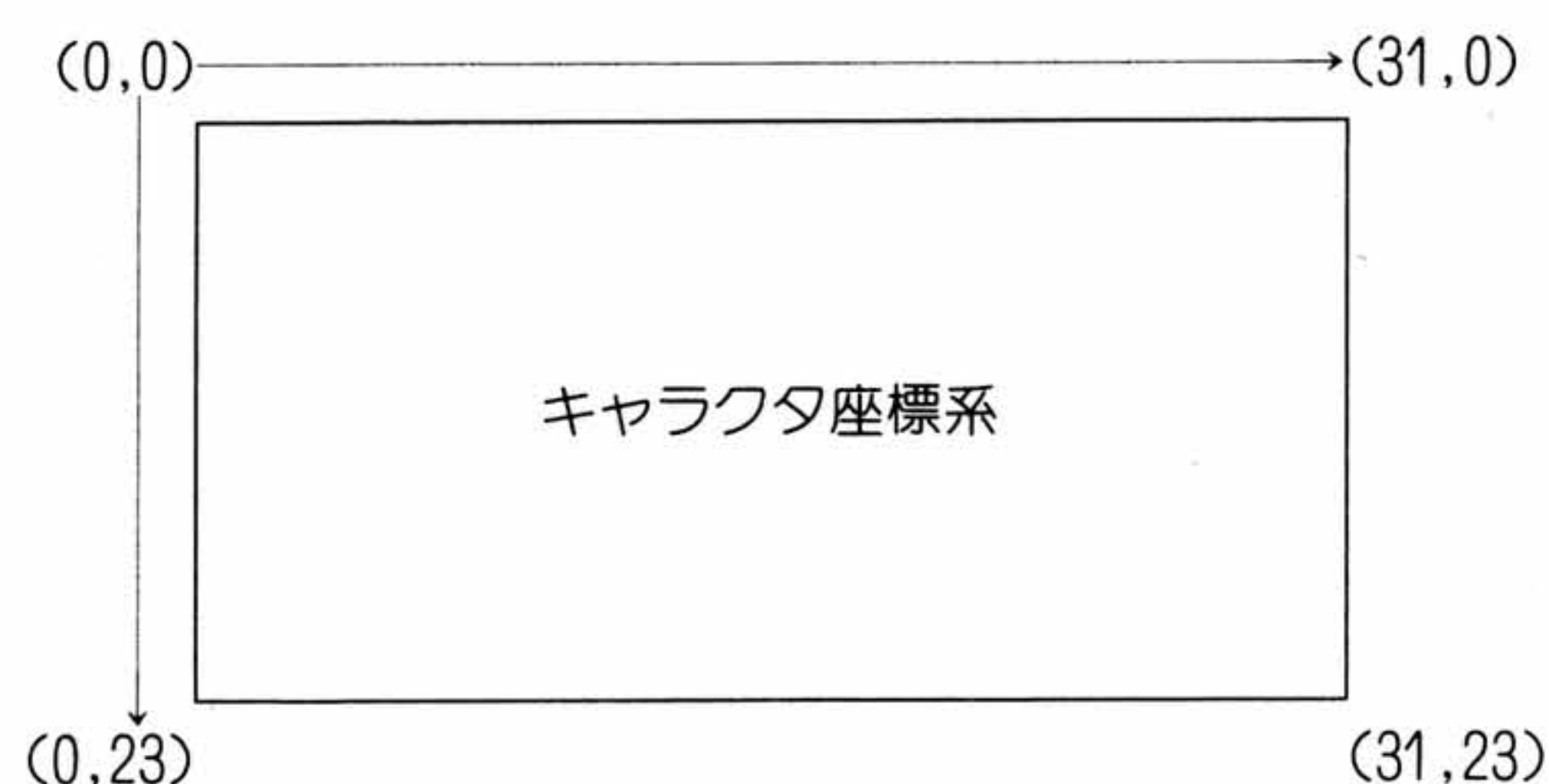
※SCREEN 4〜8はMSX2の機能です。
SCREEN 0で80×24文字はMSX2の機能です。
SCREEN 7と8は、MSX2 のVRAM容量が128KBの機種のみ有効です。

●テキストモード（40×24）：SCREEN 0

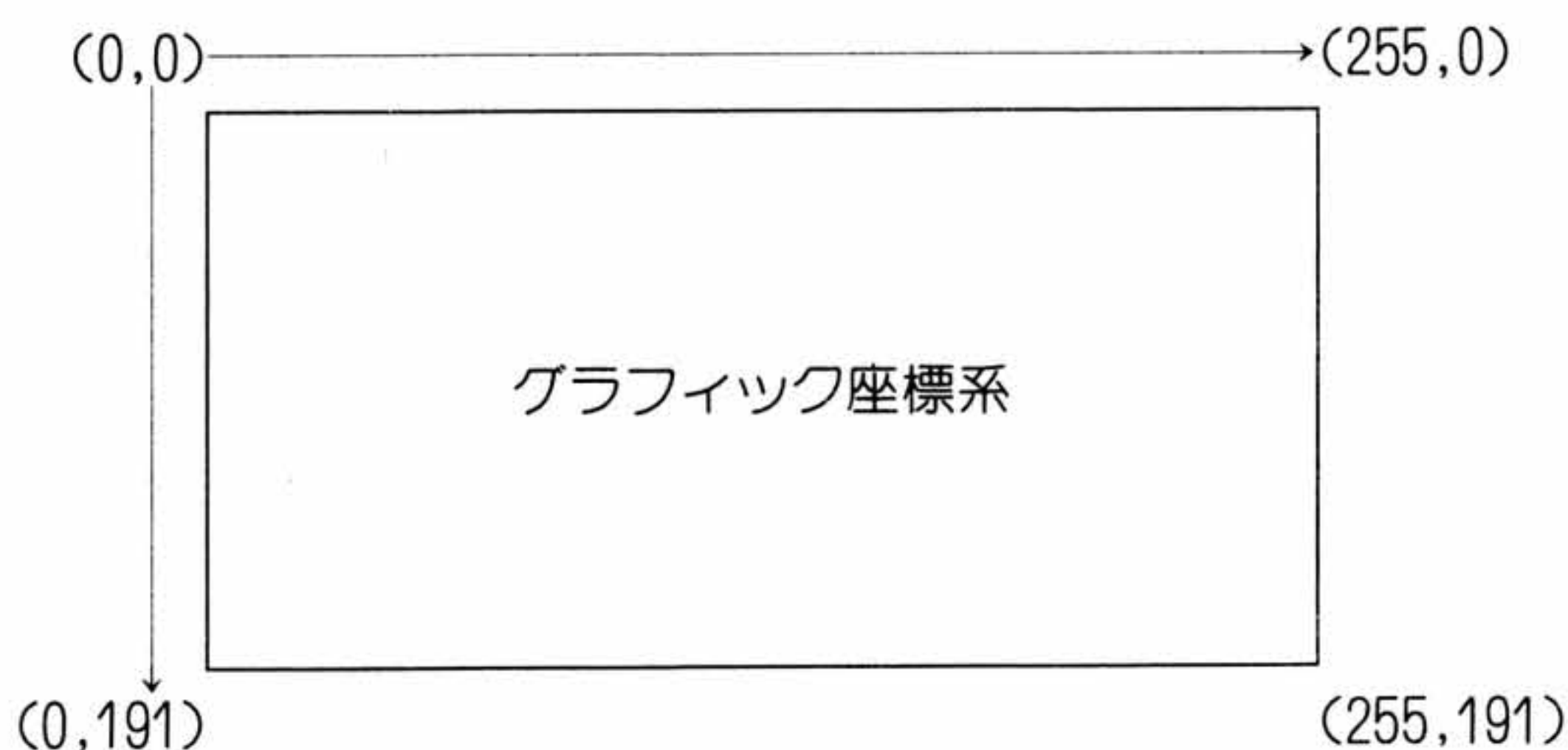
最大表示文字数は横40桁、縦24行で、表示文字のサイズは横6ドット×縦8ドットです。画面に表示する文字の桁数はWIDTH文で指定することができますが、指定がなければ39桁になっています。このモードではスプライト機能やグラフィック命令を使うことはできません（MSX2では80×24も可能）。

(0,0)→(39,0)
(79.0)：MSX2
キャラクタ座標系
(0,23)
(39,23)
(79.23)：MSX2

●テキストモード（32×24）：SCREEN 1

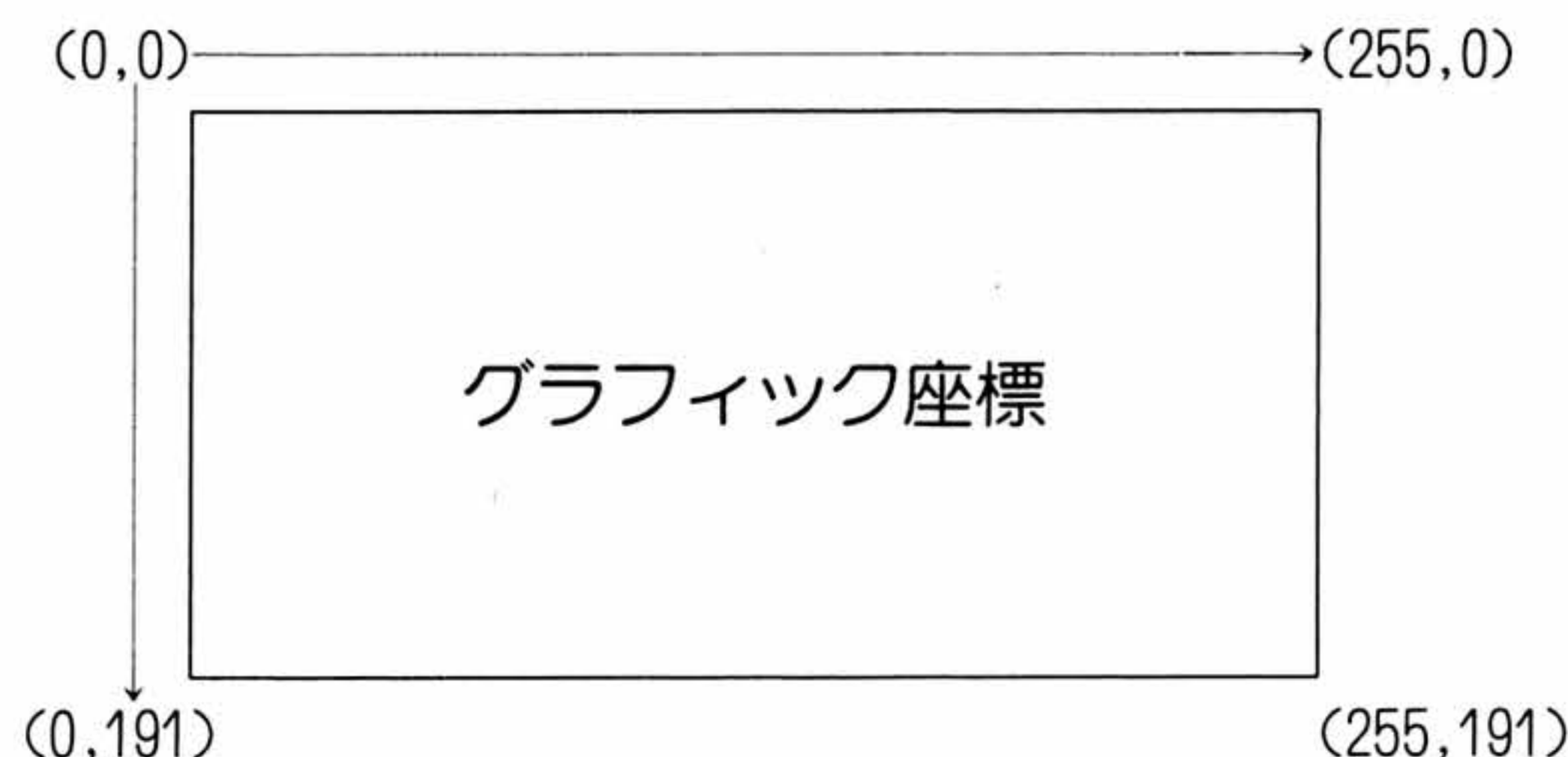
最大表示文字数は横32桁、縦24行で、表示文字サイズは横8ドット×縦8ドットです。画面に表示する文字の桁数はWIDTH文で指定することができますが、指定がなければ29桁になっています。このモードではスプライト機能は使えますが、グラフィック命令を使うことはできません。

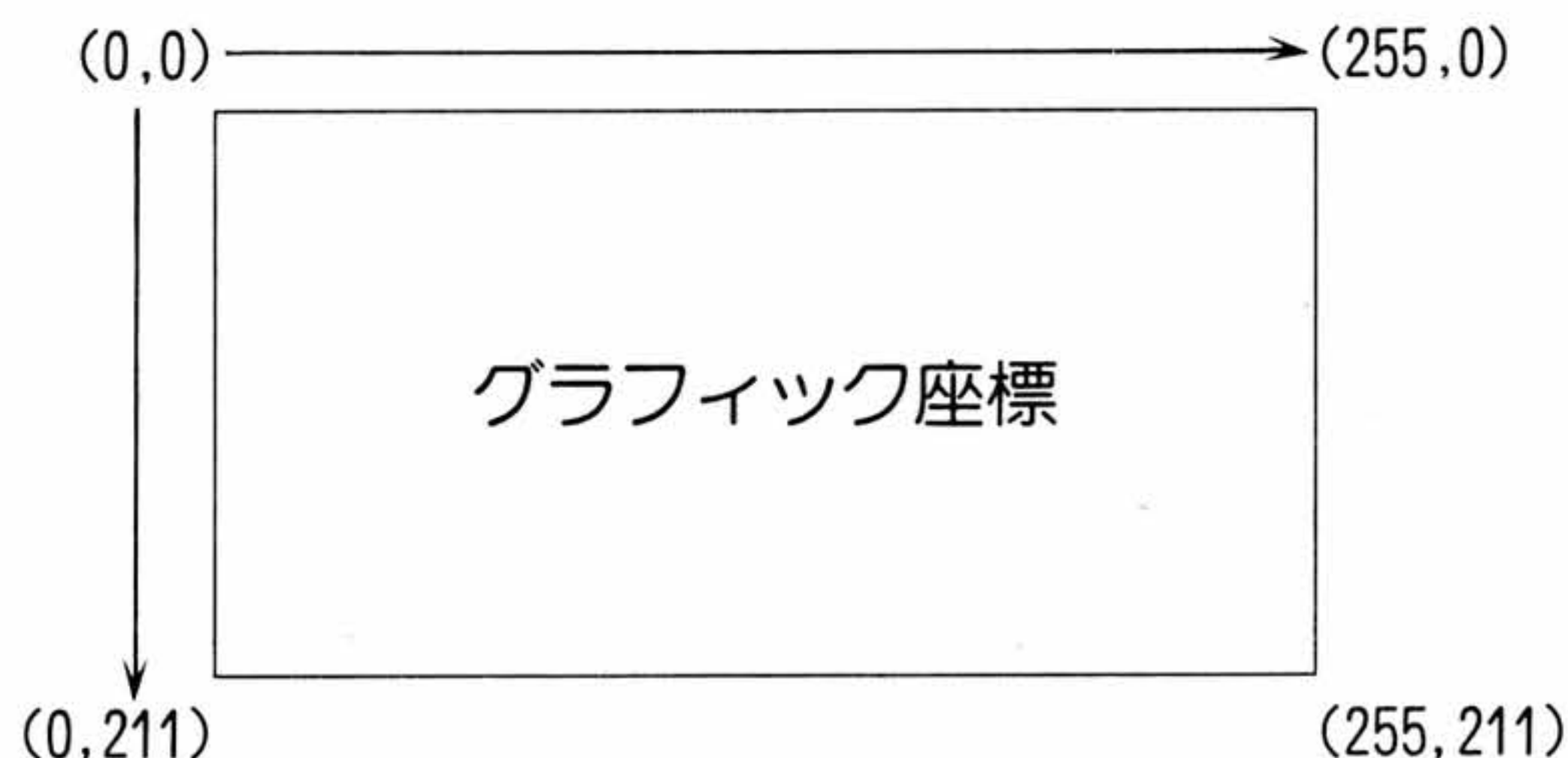
(0,0)→(31,0)
キャラクタ座標系
(0,23)
(31,23)

●高解像度グラフィックモード：SCREEN 2と4

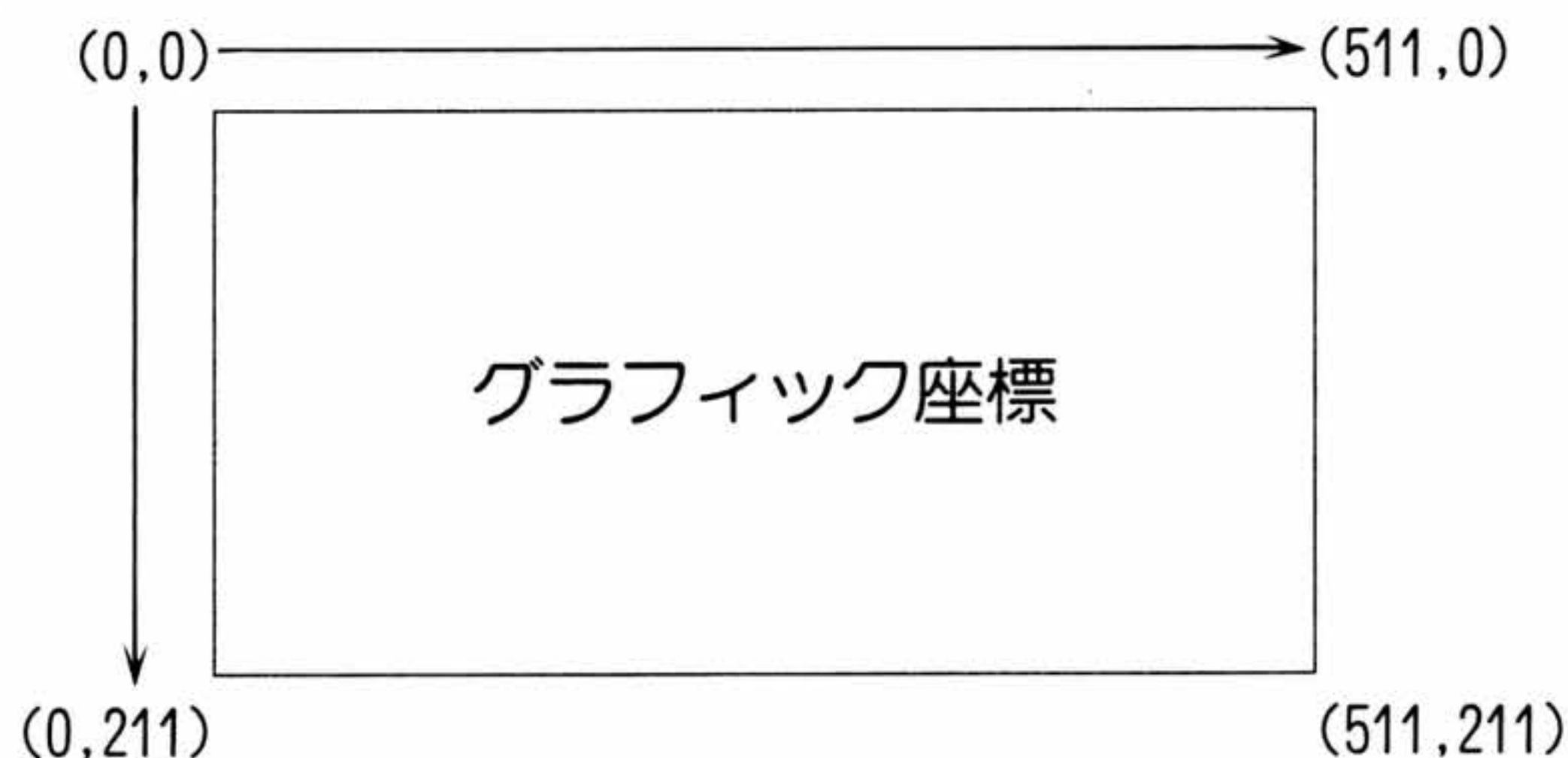
256×192ドットの解像度のグラフィック画面モード。解像度が高いので細い点や線を描くことができます。このモードではPRINT文やINPUT文などを使うことはできません。また、画面に文字を表示する場合には、OPEN文で“GRP:”ファイルを開き、PRINT #文で文字を出力します。

(0,0)→(255,0)
グラフィック座標系
(0,191)
(255,191)

高解像度グラフィックモードでは、画面を（0，0）から右へ8ドットごとに分割して表示色を管理しています。分割された8ドットごとに、背景色を含めて2色まで使うことができます。

● マルチカラーモード：SCREEN 3

64×48ブロックの解像度のグラフィック画面モード。
点を描く最小単位が4×4ドットのブロック単位なので高解像度グラフィックモードのように細い点や線を描くことはできませんが、1ブロックごとに色を使い分けることができます。
このモードではPRINT文やINPUT文などを使うことはできません。画面に文字を表示する場合には、OPEN文で"GRP:"ファイルを開き、PRINT#文で文字を出力します。(文字は約4倍に拡大されて表示されます)

● ビットマップグラフィックモード：SCREEN 5と8

256×212ドットの解像度のグラフィック画面モード。SCREEN 5では1ドットごとに16色(512色の中から選択)の中から自由に描けます。SCREEN 8では1ドットごとに、256色の固定色の中から自由に描けます。

● ビットマップグラフィックモード：SCREEN 6と7

512×212ドットの解像度のグラフィック画面モード。SCREEN 6では4色/512色が指定でき、SCREEN 7では16色/512色が指定できます。いづれも1ドットごとに色を設定できます。

5.1.14 カラーコード

MSX BASICでは次の16色を使うことができます。色はそれぞれのカラーコードで指定します。
また、MSX2では、各カラーコードの色あいを変えることもできます(カラーパレット機能)。

- 0 — 透明(周辺色と同じ色)
- 1 — 黒
- 2 — 緑
- 3 — 明るい緑
- 4 — 暗い青
- 5 — 明るい青
- 6 — 暗い赤
- 7 — 水色
- 8 — 赤
- 9 — 明るい赤
- 10 — 黄
- 11 — 明るい黄
- 12 — 暗い緑
- 13 — 紫
- 14 — 灰
- 15 — 白

5.2 資料

5.2.1 予約語表

変数名として使用できません。

ABS
AND
ASC
ATN
ATTR$
AUTO
BASE
BEEP
BIN$
BLOAD
BSAVE
CALL
CDBL
CHR$
CINT
CIRCLE
CLEAR
CLOAD
CLOSE
CLS
CMD
COLOR
CONT
COPY
COS
CSAVE
CSNG
CSRLIN
CVD
CVI
CVS
DATA
DEF
DEFDBL
DEFINT
DEFSNG
DEFSTR
DELETE
DIM
DRAW
DSKF
DSKI$
DSKO$
ELSE
END
EOF
EQV
ERASE
ERL
ERR
ERROR
EXP
FIELD
FILES
FIX
FN
FOR
FPOS
FRE
GET
GO TO
GOSUB
GOTO
HEX$
IF
IMP
INKEY$
INP
INPUT
INSTR
INT
IPL
KEY
KILL
LEFT$
LEN
LET
LFILES
LINE
LIST
LLIST
LOAD
LOC
LOCATE
LOF
LOG
LPOS
LPRINT
LSET
MAX
MERGE
MID$
MKD$
MKI$
MKS$
MOD
MOTOR
NAME
NEW
NEXT
NOT
OCT$
OFF
ON
OPEN
OR
OUT
PAD
PAINT
PDL
PEEK
PLAY
POINT
POKE
POS
PRESET
PRINT
PSET
PUT
READ
REM
RENUM
RESTORE
RESUME
RETURN
RIGHT$
RND
RSET
RUN
SAVE
SCREEN
SET
SGN
SIN
SOUND
SPACE$
SPC(
SPRITE
SQR
STEP
STICK
STOP
STR$
STRIG
STRING$
SWAP
TAB(
TAN
THEN
TIME
TO
TROFF
TRON
USING
USR
VAL
VARPTR
VDP
VPEEK
VPOKE
WAIT
WIDTH
XOR

5.2.2 キャラクタコード表

コード(10進)	コード(16進)	キャラクタ	コード(10進)	コード(16進)	キャラクタ	コード(10進)	コード(16進)	キャラクタ	コード(10進)	コード(16進)	キャラクタ
0	00	↑	32	20		64	40	@	96	60	`
1	01		33	21	!	65	41	A	97	61	a
2	02		34	22	”	66	42	B	98	62	b
3	03		35	23	#	67	43	C	99	63	c
4	04		36	24	$	68	44	D	100	64	d
5	05		37	25	%	69	45	E	101	65	e
6	06		38	26	&	70	46	F	102	66	f
7	07		39	27	'	71	47	G	103	67	g
8	08		40	28	(	72	48	H	104	68	h
9	09	コ	41	29	)	73	49	I	105	69	i
10	0A		42	2A	*	74	4A	J	106	6A	j
11	0B	ン	43	2B	+	75	4B	K	107	6B	k
12	0C	ト	44	2C	,	76	4C	L	108	6C	l
13	0D		45	2D	−	77	4D	M	109	6D	m
14	0E	ロ	46	2E	.	78	4E	N	110	6E	n
15	0F	ー	47	2F	/	79	4F	O	111	6F	o
16	10	ル	48	30	0	80	50	P	112	70	p
17	11		49	31	1	81	51	Q	113	71	q
18	12	コ	50	32	2	82	52	R	114	72	r
19	13	ー	51	33	3	83	53	S	115	73	s
20	14		52	34	4	84	54	T	116	74	t
21	15	ド	53	35	5	85	55	U	117	75	u
22	16		54	36	6	86	56	V	118	76	v
23	17		55	37	7	87	57	W	119	77	w
24	18		56	38	8	88	58	X	120	78	x
25	19		57	39	9	89	59	Y	121	79	y
26	1A		58	3A	:	90	5A	Z	122	7A	z
27	1B		59	3B	;	91	5B	[	123	7B	{
28	1C		60	3C	<	92	5C	¥	124	7C	|
29	1D		61	3D	=	93	5D	]	125	7D	}
30	1E		62	3E	>	94	5E	^	126	7E	~
31	1F	↓	63	3F	?	95	5F	—	127	7F	(DEL)

コード(10進)	コード(16進)	キャラクタ	コード(10進)	コード(16進)	キャラクタ	コード(10進)	コード(16進)	キャラクタ	コード(10進)	コード(16進)	キャラクタ
128	80	♠	160	A0		192	C0	タ	224	E0	た
129	81	♥	161	A1	。	193	C1	チ	225	E1	ち
130	82	♣	162	A2	「	194	C2	ツ	226	E2	つ
131	83	◆	163	A3	」	195	C3	テ	227	E3	て
132	84	○	164	A4	、	196	C4	ト	228	E4	と
133	85	●	165	A5	・	197	C5	ナ	229	E5	な
134	86	を	166	A6	ヲ	198	C6	ニ	230	E6	に
135	87	ぁ	167	A7	ァ	199	C7	ヌ	231	E7	ぬ
136	88	ぃ	168	A8	ィ	200	C8	ネ	232	E8	ね
137	89	ぅ	169	A9	ゥ	201	C9	ノ	233	E9	の
138	8A	ぇ	170	AA	ェ	202	CA	ハ	234	EA	は
139	8B	ぉ	171	AB	ォ	203	CB	ヒ	235	EB	ひ
140	8C	ゃ	172	AC	ャ	204	CC	フ	236	EC	ふ
141	8D	ゅ	173	AD	ュ	205	CD	ヘ	237	ED	へ
142	8E	ょ	174	AE	ョ	206	CE	ホ	238	EE	ほ
143	8F	っ	175	AF	ッ	207	CF	マ	239	EF	ま
144	90		176	B0	ー	208	D0	ミ	240	F0	み
145	91	あ	177	B1	ア	209	D1	ム	241	F1	む
146	92	い	178	B2	イ	210	D2	メ	242	F2	め
147	93	う	179	B3	ウ	211	D3	モ	243	F3	も
148	94	え	180	B4	エ	212	D4	ヤ	244	F4	や
149	95	お	181	B5	オ	213	D5	ユ	245	F5	ゆ
150	96	か	182	B6	カ	214	D6	ヨ	246	F6	よ
151	97	き	183	B7	キ	215	D7	ラ	247	F7	ら
152	98	く	184	B8	ク	216	D8	リ	248	F8	り
153	99	け	185	B9	ケ	217	D9	ル	249	F9	る
154	9A	こ	186	BA	コ	218	DA	レ	250	FA	れ
155	9B	さ	187	BB	サ	219	DB	ロ	251	FB	ろ
156	9C	し	188	BC	シ	220	DC	ワ	252	FC	わ
157	9D	す	189	BD	ス	221	DD	ン	253	FD	ん
158	9E	せ	190	BE	セ	222	DE	゛	254	FE	
159	9F	そ	191	BF	ソ	223	DF	゜	255	FF	

5.2.3 グラフィックキャラクターコード表

コード（10進）	コード（16進）	キャラクタ	コード（10進）	コード（16進）	キャラクタ
0	00		16	10	π
1	01	月	17	11	┴
2	02	火	18	12	┬
3	03	水	19	13	┤
4	04	木	20	14	├
5	05	金	21	15	┼
6	06	土	22	16	│
7	07	日	23	17	─
8	08	年	24	18	┌
9	09	円	25	19	┐
10	0A	時	26	1A	└
11	0B	分	27	1B	┘
12	0C	秒	28	1C	╳
13	0D	百	29	1D	大
14	0E	千	30	1E	中
15	0F	万	31	1F	小

ここに掲げるグラフィックキャラクタをキーボードから入力するとCHR$(1)＋CHR$(グラフィックキャラクタコード＋64)の2バイトが入力されます。また、PRINT文やLPRINT文でグラフィックキャラクタを出力すると、(例:PRINT"月")同じように、CHR$(1)＋CHR$（グラフィックキャラクタコード＋64)の2バイトが自動的に出力されます。ここで1文字目のCHR$(1)は、後の文字がグラフィックキャラクタであることを表すコントロールコードです。CHR$関数を使ってグラフィックキャラクタを出力する場合はCHR$(1)＋CHR$(グラフィックキャラクタコード＋64)で出力してください。

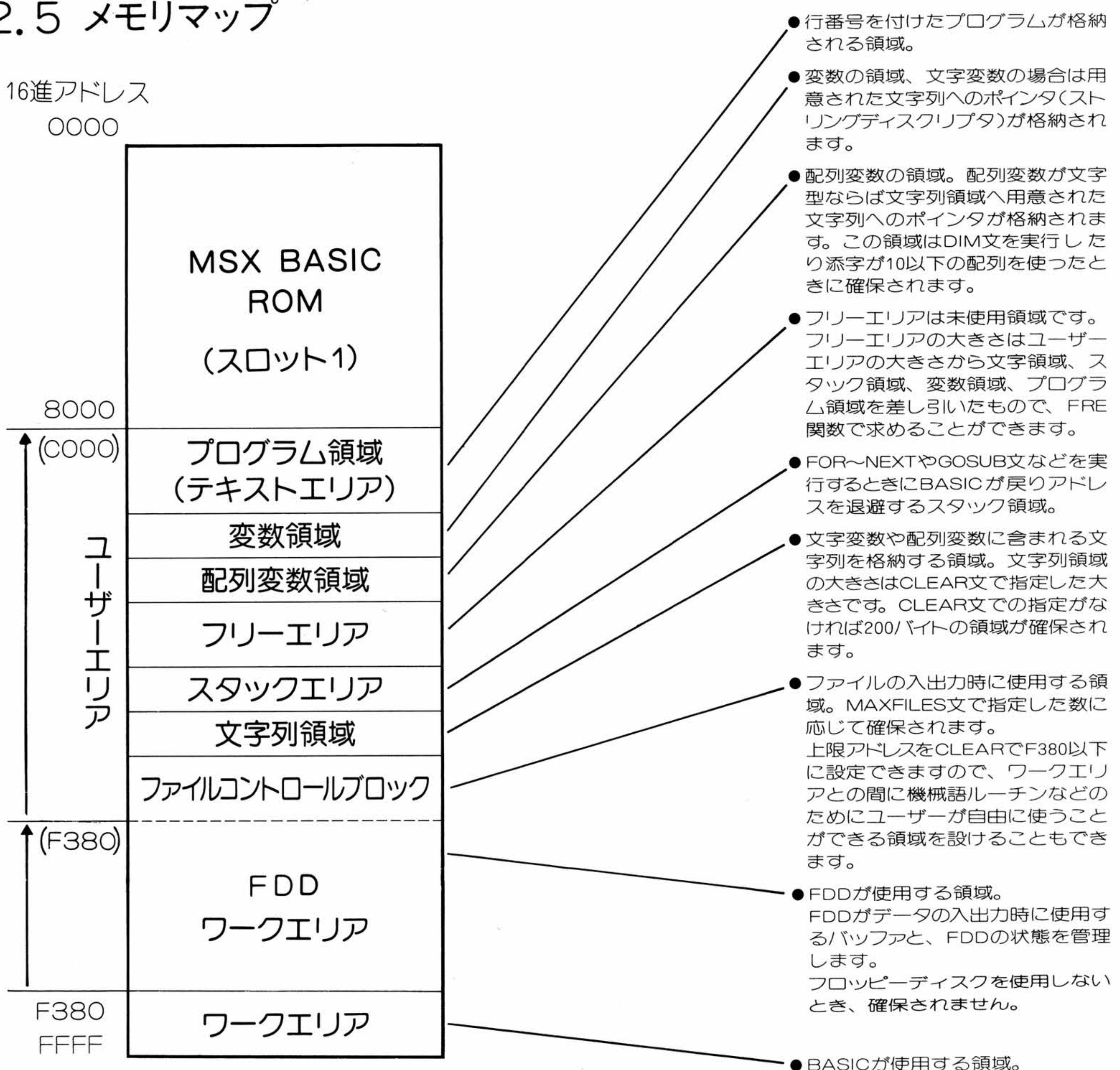
5-2

5.2.4 コントロールコード表

コード (10進)	コード (16進)	機　　　　能	対応キー
0	00		
1	01	グラフィックキャラクタの入出力時のヘッダ	CTRL + A
2	02	カーソルを直前の語の先頭へ移動する	CTRL + B
3	03	入力待ち状態を終了する	CTRL + C
4	04		CTRL + D
5	05	カーソル以下を削除	CTRL + E
6	06	カーソルを次の語の先頭へ移動	CTRL + F
7	07	スピーカを鳴らす(BEEP文と同じ)	CTRL + G
8	08	カーソルの1つ前の文字を削除する	CTRL + H , BS
9	09	次の水平タブ位置へ移動	CTRL + I , TAB
10	0A	行送り(ラインフィード)	CTRL + J
11	0B	カーソルをホームポジション(左上)に戻す	CTRL + K
12	0C	画面をクリアし、カーソルをホームポジションに戻す	CTRL + L
13	0D	カーソルを左端に戻す(キャリッジリターン)	CTRL + M , RETURN
14	0E	カーソルを行末へ移動	CTRL + N
15	0F		CTRL + O
16	10		CTRL + P
17	11		CTRL + Q
18	12	挿入モードのON/OFFスイッチ	CTRL + R , INS
19	13		CTRL + S
20	14		CTRL + T
21	15	一行を画面から削除	CTRL + U
22	16		CTRL + V
23	17		CTRL + W
24	18		CTRL + X , SELECT
25	19		CTRL + Y
26	1A		CTRL + Z
27	1B		CTRL + [, ESC
28	1C	カーソルを右へ移動	CTRL + ¥ , →
29	1D	カーソルを左へ移動	CTRL +] , ←
30	1E	カーソルを上へ移動	CTRL + ^ , ↑
31	1F	カーソルを下へ移動	CTRL + _ , ↓
127	7F	カーソルの指す文字を削除	DEL
—		プログラムの実行、入力待ちの状態を終了する	CTRL + STOP

(注)　対応キーの“，”は、2つの操作が同じ働きをすることを示しています。

例)　CTRL + I は、TAB と同じ

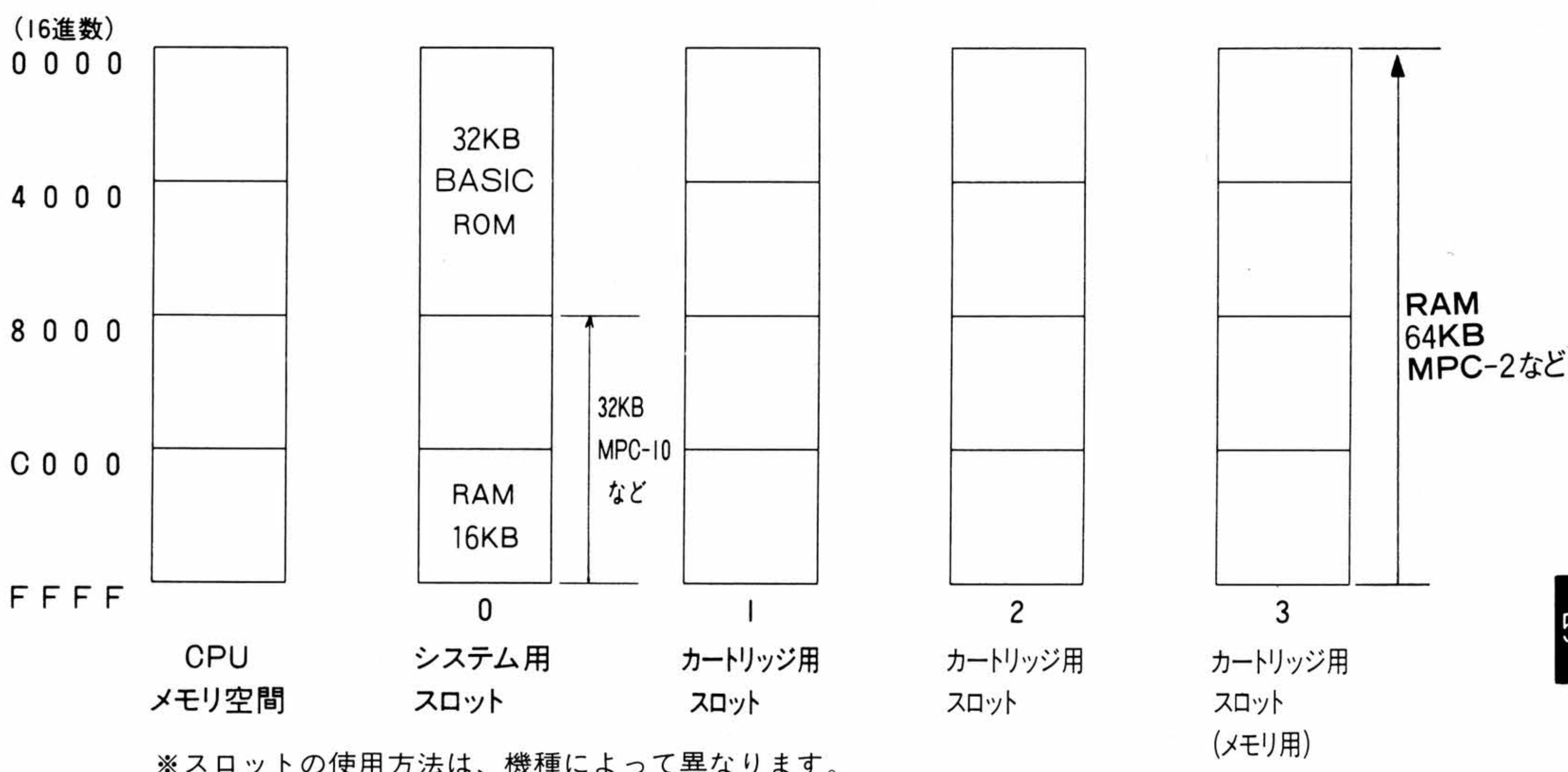
5.2.5 メモリマップ

●FDDワークエリアの下限値は、フロッピーディスクの接続状態によって異なります。

ユーザーエリアは16KBRAM実装の場合$(C000)_{16}$～$(F37F)_{16}$、また、32KBRAM実装の場合$(8000)_{16}$～$(F37F)_{16}$です。

また、64KBRAM実装の場合でも、ユーザーエリアは$(8000)_{16}$～$(F37F)_{16}$です。

※スロットの使用方法は、機種によって異なります。

5.2.6 I/Oマップ

I/O アドレス	
00	
80	※RS-232C
90	※プリンタ
98	VDP
A0	PSG
A8	PPI
B0	
B8	※LPN
BB	
D0	※FDC
D8	※漢字ROM
E0	
FF	

I/Oアドレス	R/W	内　　容	備　考
& H90	W	ストローブ出力(b0)	ラッチ出力
	R	ステータス入力(b1)	BUSY'1'
& H91	W	プリントデータ	ラッチ出力
& H98	W	VRAMへのデータライト	9918A
	R	VRAMからのデータリード	相当品
& H99	W	コマンド、アドレスセット	
	R	ステータスリード	
& HA0	W	アドレスラッチ	AY-3-8910
& HA1	W	データライト	相当品
& HA2	R	データリード	
& HA8	W	ポートAデータライト	8255A
	R	ポートAデータリード	相当品
& HA9	W	ポートBデータライト	
	R	ポートBデータリード	
& HAA	W	ポートCデータライト	
	R	ポートCデータリード	
& HAB	W	モードセット	
& HAC ∫ & HAF	—	MSX-ENGINE	
& HB0	W	アドレス　A0～A7	拡張メモリー
& HB1	W	アドレス　A8～A10、A13～A15	
& HB2	W	アドレス　A11～A12	
	R	データリード　D0～D7	
& HB3	W	モードセット	
& HB4	W	アドレスラッチ	カレンダー
& HB5	W	データライト	クロック
	R	データリード	MSX2
& HB8	R	ライトペン垂直アドレス	
	W	フラッシュ開始垂直アドレス	
& HB9	R	ライトペン水平アドレス	
	W	フラッシュ開始水平アドレス	
& HBA	R	ライトペンSW、ライトペン信号	
	W	フラッシュ開始水平アドレス	
& HBB	W	フラッシュサイズ、信号リセット	

※印の入出力番地は拡張機器のために用意されています。

I/Oアドレス	R/W	内　　容	備　考
&HBC &HBD &HBE &HBE	—	ポートA ポートB ポートC モードセット	VHD コントロール
&HC0 &HC1	—	MSX-Audio	MSX2
&HD8 &HD9	W W R	下位アドレスライト(bit5〜bit0) 上位アドレスライト(bit5〜bit0) データリード　(bit7〜bit0)	
&HF5	—	システムコントロール	MSX2
&HF6	—	カラーバス　I/O	MSX2
&HF7		AVコントロール	MSX2
&HFC &HFD &HFE &HFF	R/W	ページ 0 ページ 1 ページ 2 ページ 3 } メモリ・マップ レジスタ	メモリ・マップ MSX2

5.2.7 キーボード表

GRAPH キーを押しながらキーを押すと、下図のようなグラフィック記号がキー入力されます。

5.2.8 エラーメッセージ表

メッセージ	エラーコード	説　　明
Bad allocation torble	60	ディスクがフォーマットされていない。
Bad drive name	62	ドライブ名の指定が間違っている。
Bad file mode	61	シーケンシャルファイルに対してPUT，GET文、あるいはLOF関数を使用した。 ランダムファイルに対してLOAD文を使用した。 OPEN文のファイルモードとしてI，O，A，R以外を設定した。
Bad file name	56	ファイル名が適当でない。
Bad file number	52	オープンしていない（開いていない）ファイル番号を指定した。指定したファイル番号がMAXFILES文で指定した数を越えている。
Bad sector number	63	セクタ番号の指定が間違っている。
Can't CONTINUE	17	CONTコマンドでプログラムの実行を再開できない。 エラーが発生して終了したプログラムは再開できません。 また、プログラムの1部を書き換えたり、CLEAR文などを実行すると、次に実行すべきポインタが壊れてプログラムを再開することはできなくなります。
Device I/O error	19	カセットレコーダ、プリンタ、テレビなどへの入出力中にエラーが発生した。
Direct statement	57	アスキー形式のプログラムファイルをロード中、プログラム以外のデータがあった。
Disk full	66	ディスク上の記憶域はすべて使用されている。
Disk I/O error	69	ディスクの入出力操作において、回復できないエラーが発生した。
Disk offline	70	ディスクがつながっていない。
Disk write protected	68	書き込み禁止のディスクに書き込みを行った。
Division by zero	11	0による除算を行おうとした。または、0に対して負のべき乗を行った。

メッセージ	エラーコード	説明
FIELD overflow	50	FIELD文で定義したフィールドサイズの合計が256バイトを越えた。
File already exists	65	NAME文の新しいファイル名として指定された名前が、ディスク上に登録されているファイル名と重複した。
File already open	54	OPEN文で指定したファイル番号は、他のファイルで使用中です。 現在開いているファイルに対してKILL文を実行した。
File not found	53	指定されたファイルが見つからない。
File not OPEN	59	OPEN文で開いていないファイルに対して入出力を行おうとした。
File still open	64	ファイルのCLOSEを忘れている。
Illegal direct	12	ダイレクトモードでは実行できないステートメントをダイレクトモードで実行しようとした。
Illegal function call	5	ステートメントや関数の使い方が誤っている。引数などが指定可能な範囲を越えている。
Input past end	55	ファイル中のデータをすべて読み込んだ後、さらに入力文を実行した。ファイル中のデータをすべて読み込んだかどうかはEOF関数で調べてください。
Internal error	51	BASIC(またはDisk-BASIC)内部でエラーが発生した。
Line buffer overflow	25	入力されたデータが許されている文字数を越えた。
Missing operand	24	文中に必要なパラメータが指定してない。
NEXT without FOR	1	NEXT文に対応するFOR文がない。
No RESUME	21	エラー処理ルーチンにRESUME文がない。
Out of DATA	4	READ文で読み出せるデータがない。
Out of memory	7	プログラムが長すぎたり、配列が大きすぎたりなどの原因でメモリが足りなくなった。

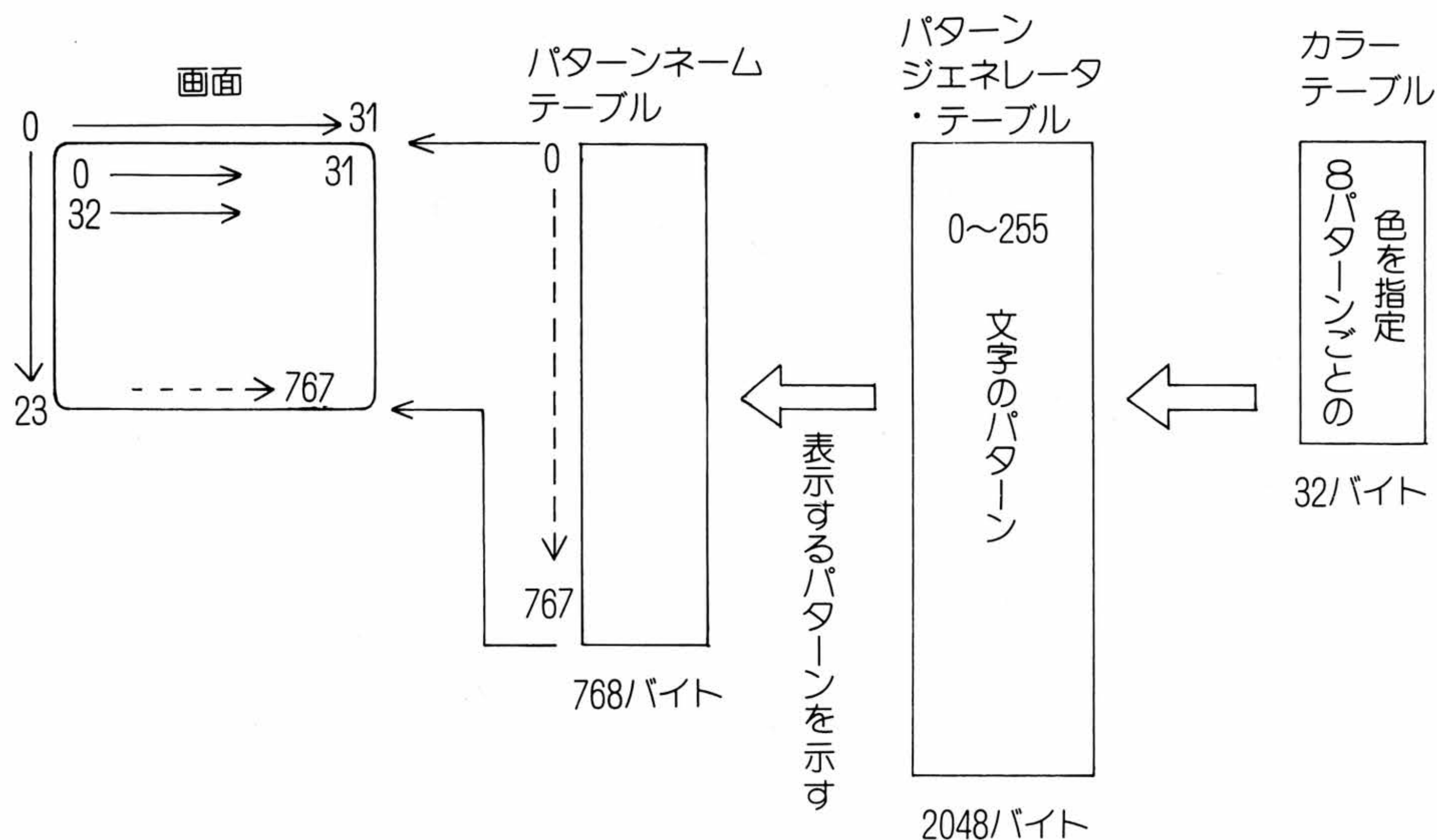
メッセージ	エラーコード	説　　明
Out of string space	14	文字列変数用のメモリが足りなくなった。
Overflow	6	演算結果や、入力された数値が型に応じた許容範囲を超えた。
Redimensioned array	10	配列変数を2重に定義しようとした。または、配列変数に添字の最大値10が割当てられた後で配列を定義しようとした。
Rename across disk	71	異なるドライブ間でNAME文を実行した。
RESUME without error	22	エラー処理ルーチン以外でRESUME文を使用した。
RETURN without GOSUB	3	GOSUB文とRETURN文が正しく対応していない。
Sequential I/O only	58	シーケンシャルファイルに対してランダム入出力を行おうとした。
String formula too complex	16	指定した文字列が複雑すぎる。文字式を2つ以上に分けてください。
String too long	15	文字列が255文字より大きい。
Subscript out of range	9	指定した配列変数の添字の値が、DIM文で定義した配列変数の添字の最大値より大きい。
Syntax error	2	文の書き方(構文)を誤っている。(かっこが対応していない、コマンド・ステートメントの綴りを誤っている、区切り記号がないなど)
Too many files	67	ディスケットのファイル数が112を越えた。(新しいファイルを作成しようとした)
Type mismatch	13	数式や数値関数に文字列を使ったり、文字式や文字関数に数値を使うなど、データの形式が一致しない。
Undefined line number	8	GOTO文、GOSUB文、IF文などで指定した行番号がプログラム中にない。
Undefined user function	18	DEF FNで定義していないユーザー定義関数を使っている。

メ ッ セ ー ジ	エラーコード	説　　明
Unprintable error	23 26～49 72～255	メッセージが定義されていないエラー
Verify error	20	カセットにセーブされたプログラムはメモリ上のプログラムと内容が異なる。 プログラムのセーブをやり直してください。

5. 2. 9 フォントについて

フォント（Font：活字の一そろい）とは、画面に表示される文字や記号の形（パターン）のことです。
MSXでは、SCREEN 0～4の画面に表示する文字や記号をパターンとして記憶しています〔SCREEN 5～8の文字や記号は、点の集りとして記憶しています（ビットマップ方式）〕。
SCREEN 0～4の画面に表示される文字や記号は、パターンジェネレータテーブルとパターンネームテーブル、そしてカラーテーブルの３つの情報によって構成されます。
（SCREEN 1～4については、この他にスプライトの情報も必要です）

■データの構成

（SCREEN 1の例）

●パターンジェネレータ・テーブル

ビデオRAMに256種類のパターンの形(文字など)を記憶します。１つの文字を８バイトのメモリで構成し、全体で2048バイトとなっています。パターンには0～255の番号（パターン名称といいます）がつけられています（キャラクターコード表の番号と同じ）。

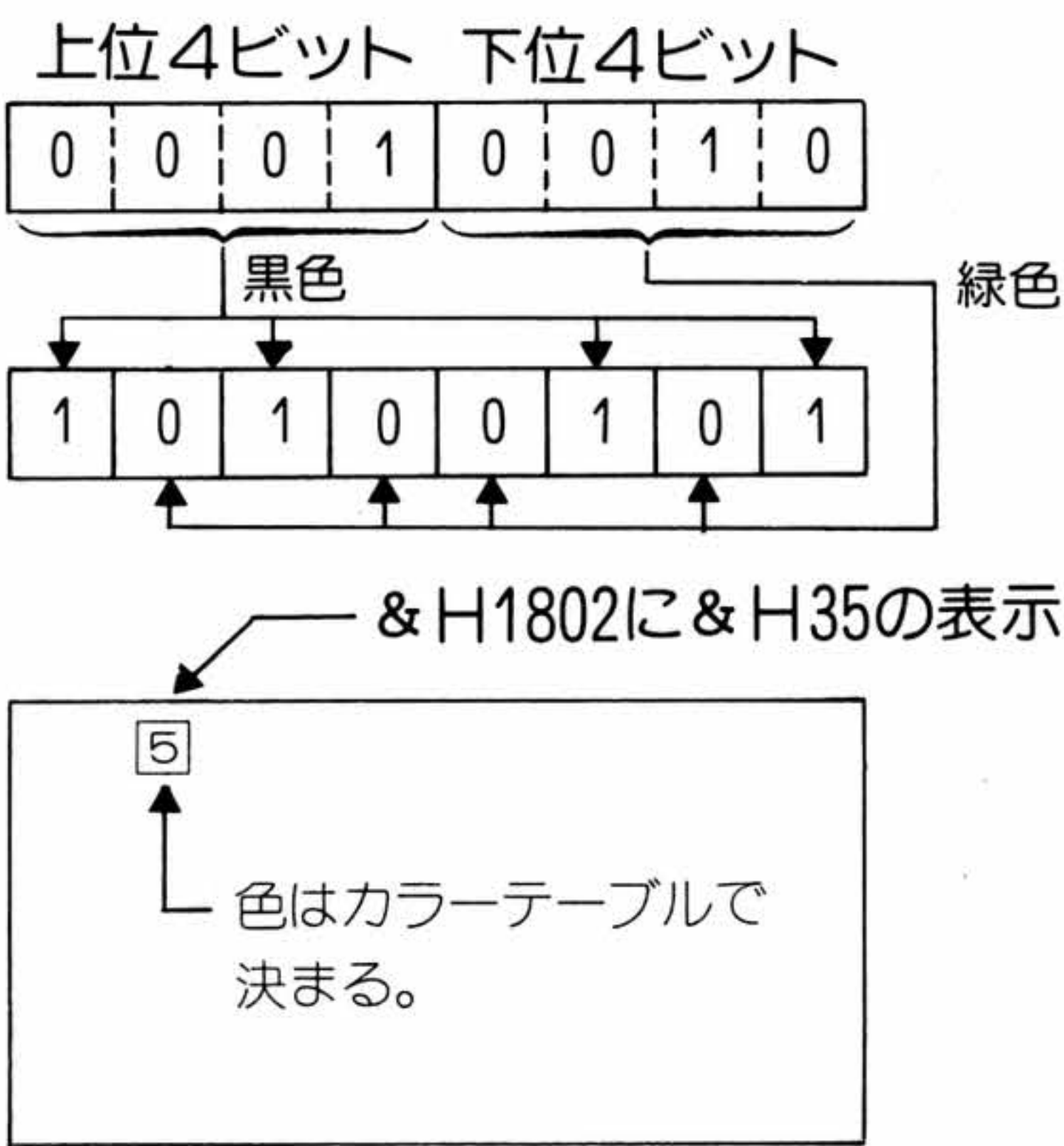
●パターンネームテーブル

画面（32文字×24行）に対応した768のメモリで、画面の表示位置に対応しています。画面に文字を表示するには、パターンネームテーブルの画面に対応したメモリ番地にパターンの番号（パターン名称）を入れます。

例　VPOKE，&H1805，&H35

●カラーテーブル

パターン名称の色を設定します。複数個のパターンを一括して設定します。(表示する画面の場所を指定するわけではありません)。

カラーテーブルには、パターン名称の番号（0〜256）を8つ1組みにして色を決めます（256÷8=32、32バイトのメモリ)。つまり、パターンの番号が0〜7のパターンは、パターンの表示する部分（ビット「1」）は何色で、表示しない部分（ビット「0」）は何色とします。
カラーテーブルの1バイト（＝8ビット）は8つのパターンの色を決め、上位4ビットがビット「1」の色を、下位4ビットがビット「0」の色を指定します。

テーブル表

テーブルの名前	各SCREENでのテーブルの開始番地			
	SCREEN0	SCREEN1	SCREEN2,4	SCREEN3
パターンネームテーブル	0000H	1800H	1800H	0800H
カラーテーブル	--------------	2000H	2000H	--------------
パターン ジェネレータ・テーブル	0800H	0000H	0000H	0000H
スプライト属性テーブル	--------------	1B00H	1B00H	1B00H
スプライト・ジェネレータ・テーブル	--------------	3800H	3800H	3800H

■パターンデータの読み込み/書き替え

文字の形（フォント）は、パターンジェネレータ・テーブルに次のように記録されています。

例　小文字のｋ（パターン番号 107=6BH, SCREEN 1のとき）

●パターンの読み込み（SCREEN 1のとき）

次のプログラムで、「ｋ」のパターンデータが読み込めます

```
10 C=&H6B
20 FOR T=0 TO 7
30 A=VPEEK(C*8+T)
40 PRINT BIN$(A)
50 NEXT T
```

●パターンの書き替え（ｋを>にする）

次のプログラムで画面に表示する文字の形を変えることができます

```
10 C=&H6B:D=C*8
20 FOR T=D TO D+7
30 READ A$:A=VAL("&h"+A$)
40 VPOKE T,A
50 NEXT T
60 DATA C0,60,30,18,30,60,C0,00
```

（SCREENを切換えると、パターンジェネレータ・テーブルは再設定されます。）

■パターンの色を変えるには（8つ1組み）

次の命令を実行すると0～7の数字の色が、文字が白色で背景色が黒となります

VPOKE &H2006, &HF1

- &H2006 … 48～55番のパターン（6×8=48）
- &H1=1　カラーコードは黒色
- &HF=15　カラーコード15は白色

サンプルプログラム　SCREEN1で表を作る

```
10 SCREEN 1:COLOR 15,4
20 FOR T=&H400 TO &H4FF
30 VPOKE T,&HFF
40 NEXT T
50 VPOKE &H2011,&H14
60 VPOKE &H2012,&H24
70 VPOKE &H2013,&HB4
80 VPOKE &H2002,&H74
90 VPOKE &H2003,&H74
100 VPOKE &H2006,&H94
110 PRINT"    |
120 PRINT"ABC|oooooo6
130 PRINT"DEF|いいいい4
140 PRINT"GHI|ああああああ7
150 PRINT"JKL|くくく3
160 PRINT"    └──────────
170 PRINT"    0 2 4 6
```

5. 2. 10 漢字 ROMコード

漢字ROM内蔵のパソコン、および漢字ROMカートリッジを使用しているパソコンでは、画面に漢字を表示させることができます。（JIS第1水準の漢字と記号など）

●MSX BASIC2での使い方

SCREEN5～8に表示するには、PUT KANJI命令を使用します。SCREEN4に表示するときには、次の「MSX BASICでの使い方」と同じ方法を使ってください。

●MSX BASICでの使い方

SCREEN2およびSCREEN4（MSX2のみ）に表示するには、次のプログラムを使います。

サンプルプログラム 8ドットごとに区切られた範囲に表示するとき

```
10 '**カンジ ヒョウジ (X,Y=0,8,.. 8 Step)**
20 SCREEN 2:DIM K(31)
30 C=15        '** カンジ の カラーコード
40 B=1         '** ハイケイショク
50 A=&H3441 '** カンジ の コード
60 X=15:Y=11 '** ヒョウジ イチ ←―――――― (Xは0～30 Yは0～22)
70 GOSUB 90
80 GOTO 80
90 '** カンジ DETA INPUT **
100 D=((A¥256)+(&H211F<A)*32)*96+((A MOD
 256)-32)+(&H301F<A)*512
110 BC=C*16+B
120 OUT &HD8,(D MOD64):OUT &HD9,(D ¥64)
130 FOR T=0 TO 31
140 K(T)=INP(&HD9):NEXT T ←―――― データの入力
150 '** カンジ OUTPUT ***
160 FOR T=0 TO 31
170 D=X*8+T+Y*256+(T ¥ 16)*240
180 VPOKE D,K(T):VPOKE D+&H2000,BC
190 NEXT T:RETURN
```

（100～120行：漢字パターンのアドレスを出力）
（160～190行：画面に表示）

サンプルプログラム 任意の位置に表示するとき（表示速度は遅くなります）

```
10 '** カンジ ヒョウジ (X=0-240,Y=0-176) **
20 SCREEN 2:DIM K(31)
30 C=15        '** カンジ の カラーコード
40 A=&H3441 '** カンジ の コード
50 X=120:Y=96 '** ヒョウジ イチ ←―――――― (Xは0～240 Yは0～176)
60 GOSUB 80
70 GOTO 70
80 '** カンジ DATA INPUT **
90 D=((A¥256)+(&H211F<A)*32)*96+((A MOD
256)-32)+(&H301F<A)*512
100 OUT &HD8,(D MOD64):OUT &HD9,(D ¥64)
110 FOR T=0 TO 31
120 K(T)=INP(&HD9):NEXT T
130 '** カンジ OUTPUT **
140 FOR T=0 TO 31:D=K(T)
150 FOR R=7 TO 0 STEP-1:N=2^R
160 XX=X-R+((T MOD 16)¥8)*8+7
170 YY=Y+(T MOD 8)+(T¥16)*8
180 IF D=>N THEN PSET(XX,YY),C:D=D-N
190 NEXT R,T:RETURN
```

（120～180行：PSET文で1つ1つの点を表示します）

5-2

●コードはすべて16進形式で表現されます。例えば、"！"は＆H212Aです。

●＆H2121はスペース（空白）のコードです。

●使用する漢字ROMによって文字の形が多少異なることがあります。

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
記号	212x			、	。	，	．	・	：	；	？	！	゛	゜	´	｀	¨
	213x	＾	￣	＿	ヽ	ヾ	ゝ	ゞ	〃	仝	々	〆	○	ー	―	‐	／
	214x	＼	～	‖	｜	…	‥	‘	’	“	”	（	）	〔	〕	［	］
	215x	｛	｝	〈	〉	《	》	「	」	『	』	【	】	＋	－	±	×
	216x	÷	＝	≠	＜	＞	≦	≧	∞	∴	♂	♀	°	′	″	℃	￥
	217x	＄	¢	£	％	＃	＆	＊	＠	§	☆	★	○	●	◎	◇	
	222x		◆	□	■	△	▲	▽	▼	※	〒	→	←	↑	↓	＝	㈱
	223x	㈲	Ⅰ	Ⅱ	Ⅲ	Ⅳ	Ⅴ	Ⅵ	Ⅶ	Ⅷ	Ⅸ	Ⅹ	g	m	cm	mm	km
	224x	cm²	m²	mℓ	mg	kg	μs	㈹	K.K.	TEL	No.	インチ		グラム	メートル	センチ	ミリ
	225x	ヘクタール	アール	ドル	トン	キロ	キログラム	キロワット	ワット	リットル	パーセント	ページ	ヤード	フィート	①	②	③
	226x	④	⑤	⑥	⑦	⑧	⑨	⑩	〘	〙		☎		∫	√	￣	⊓
	227x	｜ ｜	⊔	⌒	◡	［	￣＿	］	（	）	(CR1)	(CR2)	(禁)	(均)	(↑)	(↓)	
	232x		◄	◂	- -	- -	╎	⁞	⇨								
英・数字	233x	0	1	2	3	4	5	6	7	8	9						
	234x		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	235x	P	Q	R	S	T	U	V	W	X	Y	Z					
	236x		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
	237x	p	q	r	s	t	u	v	w	x	y	z					
ひらがな	242x		ぁ	あ	ぃ	い	ぅ	う	ぇ	え	ぉ	お	か	が	き	ぎ	く
	243x	ぐ	け	げ	こ	ご	さ	ざ	し	じ	す	ず	せ	ぜ	そ	ぞ	た
	244x	だ	ち	ぢ	っ	つ	づ	て	で	と	ど	な	に	ぬ	ね	の	は
	245x	ば	ぱ	ひ	び	ぴ	ふ	ぶ	ぷ	へ	べ	ぺ	ほ	ぼ	ぽ	ま	み
	246x	む	め	も	ゃ	や	ゅ	ゆ	ょ	よ	ら	り	る	れ	ろ	ゎ	わ
	247x	ゐ	ゑ	を	ん												
カタカナ	252x		ァ	ア	ィ	イ	ゥ	ウ	ェ	エ	ォ	オ	カ	ガ	キ	ギ	ク
	253x	グ	ケ	ゲ	コ	ゴ	サ	ザ	シ	ジ	ス	ズ	セ	ゼ	ソ	ゾ	タ
	254x	ダ	チ	ヂ	ッ	ツ	ヅ	テ	デ	ト	ド	ナ	ニ	ヌ	ネ	ノ	ハ
	255x	バ	パ	ヒ	ビ	ピ	フ	ブ	プ	ヘ	ベ	ペ	ホ	ボ	ポ	マ	ミ
	256x	ム	メ	モ	ャ	ヤ	ュ	ユ	ョ	ヨ	ラ	リ	ル	レ	ロ	ヮ	ワ
	257x	ヰ	ヱ	ヲ	ン	ヴ	ヵ	ヶ									

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ギリシャ文字	262x		Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο
	263x	Π	Ρ	Σ	Τ	Υ	Φ	Χ	Ψ	Ω							
	264x		α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο
	265x	π	ρ	σ	τ	υ	φ	χ	ψ	ω							
ロシア文字	272x		А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н
	273x	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
	274x	Ю	Я														
	275x		а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н
	276x	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э
	277x	ю	я														
ア	302x		亜	唖	娃	阿	哀	愛	挨	姶	逢	葵	茜	穐	悪	握	渥
	303x	旭	葦	芦	鯵	梓	圧	斡	扱	宛	姐	虻	飴	絢	綾	鮎	或
	304x	粟	袷	安	庵	按	暗	案	闇	鞍	杏						
イ	304x											以	伊	位	依	偉	囲
	305x	夷	委	威	尉	惟	意	慰	易	椅	為	畏	異	移	維	緯	胃
	306x	萎	衣	謂	違	遺	医	井	亥	域	育	郁	磯	一	壱	溢	逸
	307x	稲	茨	芋	鰯	允	印	咽	員	因	姻	引	飲	淫	胤	蔭	
	312x		院	陰	隠	韻	吋										
ウ	312x							右	宇	烏	羽	迂	雨	卯	鵜	窺	丑
	313x	碓	臼	渦	嘘	唄	欝	蔚	鰻	姥	厩	浦	瓜	閏	噂	云	運
	314x	雲															
エ	314x		荏	餌	叡	営	嬰	影	映	曳	栄	永	泳	洩	瑛	盈	穎
	315x	頴	英	衛	詠	鋭	液	疫	益	駅	悦	謁	越	閲	榎	厭	円
	316x	園	堰	奄	宴	延	怨	掩	援	沿	演	炎	焔	煙	燕	猿	縁
	317x	艶	苑	薗	遠	鉛	鴛	塩									
オ	317x								於	汚	甥	凹	央	奥	往	応	
	322x		押	旺	横	欧	殴	王	翁	襖	鴬	鴎	黄	岡	沖	荻	億
	323x	屋	憶	臆	桶	牡	乙	俺	卸	恩	温	穏	音				
カ	323x													下	化	仮	何
	324x	伽	価	佳	加	可	嘉	夏	嫁	家	寡	科	暇	果	架	歌	河
	325x	火	珂	禍	禾	稼	箇	花	苛	茄	荷	華	菓	蝦	課	嘩	貨
	326x	迦	過	霞	蚊	俄	峨	我	牙	画	臥	芽	蛾	賀	雅	餓	駕
	327x	介	会	解	回	塊	壊	廻	快	怪	悔	恢	懐	戒	拐	改	

5-2

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
カ	332x		魁	晦	械	海	灰	界	皆	絵	芥	蟹	開	階	貝	凱	劾
	333x	外	咳	害	崖	慨	概	涯	碍	蓋	街	該	鎧	骸	浬	馨	蛙
	334x	垣	柿	蛎	鈎	劃	嚇	各	廓	拡	撹	格	核	殻	獲	確	穫
	335x	覚	角	赫	較	郭	閣	隔	革	学	岳	楽	額	顎	掛	笠	樫
	336x	橿	梶	鰍	潟	割	喝	恰	括	活	渇	滑	葛	褐	轄	且	鰹
	337x	叶	椛	樺	鞄	株	兜	竃	蒲	釜	鎌	噛	鴨	栢	茅	萱	
	342x		粥	刈	苅	瓦	乾	侃	冠	寒	刊	勘	勧	巻	喚	堪	姦
	343x	完	官	寛	干	幹	患	感	慣	憾	換	敢	柑	桓	棺	款	歓
	344x	汗	漢	澗	潅	環	甘	監	看	竿	管	簡	緩	缶	翰	肝	艦
	345x	莞	観	諌	貫	還	鑑	間	閑	関	陥	韓	館	舘	丸	含	岸
	346x	巌	玩	癌	眼	岩	翫	贋	雁	頑	顔	願					
キ	346x												企	伎	危	喜	器
	347x	基	奇	嬉	寄	岐	希	幾	忌	揮	机	旗	既	期	棋	棄	
	352x		機	帰	毅	気	汽	畿	祈	季	稀	紀	徽	規	記	貴	起
	353x	軌	輝	飢	騎	鬼	亀	偽	儀	妓	宜	戯	技	擬	欺	犠	疑
	354x	祇	義	蟻	誼	議	掬	菊	鞠	吉	吃	喫	桔	橘	詰	砧	杵
	355x	黍	却	客	脚	虐	逆	丘	久	仇	休	及	吸	宮	弓	急	救
	356x	朽	求	汲	泣	灸	球	究	窮	笈	級	糾	給	旧	牛	去	居
	357x	巨	拒	拠	挙	渠	虚	許	距	鋸	漁	禦	魚	亨	享	京	
	362x		供	侠	僑	兇	競	共	凶	協	匡	卿	叫	喬	境	峡	強
	363x	彊	怯	恐	恭	挟	教	橋	況	狂	狭	矯	胸	脅	興	蕎	郷
	364x	鏡	響	饗	驚	仰	凝	尭	暁	業	局	曲	極	玉	桐	粁	僅
	365x	勤	均	巾	錦	斤	欣	欽	琴	禁	禽	筋	緊	芹	菌	衿	襟
	366x	謹	近	金	吟	銀											
ク	366x						九	倶	句	区	狗	玖	矩	苦	躯	駆	駈
	367x	駒	具	愚	虞	喰	空	偶	寓	遇	隅	串	櫛	釧	屑	屈	
	372x		掘	窟	沓	靴	轡	窪	熊	隈	粂	栗	繰	桑	鍬	勲	君
	373x	薫	訓	群	軍	郡											
ケ	373x						卦	袈	祁	係	傾	刑	兄	啓	圭	珪	型
	374x	契	形	径	恵	慶	慧	憩	掲	携	敬	景	桂	渓	畦	稽	系
	375x	経	継	繋	罫	茎	荊	蛍	計	詣	警	軽	頚	鶏	芸	迎	鯨
	376x	劇	戟	撃	激	隙	桁	傑	欠	決	潔	穴	結	血	訣	月	件
	377x	倹	倦	健	兼	券	剣	喧	圏	堅	嫌	建	憲	懸	拳	捲	
	382x		検	権	牽	犬	献	研	硯	絹	県	肩	見	謙	賢	軒	遣
	383x	鍵	険	顕	験	鹸	元	原	厳	幻	弦	減	源	玄	現	絃	舷
	384x	言	諺	限													

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
コ	384x				乎	個	古	呼	固	姑	孤	己	庫	弧	戸	故	枯
	385x	湖	狐	糊	袴	股	胡	菰	虎	誇	跨	鈷	雇	顧	鼓	五	互
	386x	伍	午	呉	吾	娯	後	御	悟	梧	檎	瑚	碁	語	誤	護	醐
	387x	乞	鯉	交	佼	侯	候	倖	光	公	功	効	勾	厚	口	向	
	392x		后	喉	坑	垢	好	孔	孝	宏	工	巧	巷	幸	広	庚	康
	393x	弘	恒	慌	抗	拘	控	攻	昂	晃	更	杭	校	梗	構	江	洪
	394x	浩	港	溝	甲	皇	硬	稿	糠	紅	紘	絞	綱	耕	考	肯	肱
	395x	腔	膏	航	荒	行	衡	講	貢	購	郊	酵	鉱	砿	鋼	閤	降
	396x	項	香	高	鴻	剛	劫	号	合	壕	拷	濠	豪	轟	麹	克	刻
	397x	告	国	穀	酷	鵠	黒	獄	漉	腰	甑	忽	惚	骨	狛	込	
	3A2x		此	頃	今	困	坤	墾	婚	恨	懇	昏	昆	根	梱	混	痕
	3A3x	紺	艮	魂													
サ	3A3x				些	佐	叉	唆	嵯	左	差	査	沙	瑳	砂	詐	鎖
	3A4x	裟	坐	座	挫	債	催	再	最	哉	塞	妻	宰	彩	才	採	栽
	3A5x	歳	済	災	采	犀	砕	砦	祭	斎	細	菜	裁	載	際	剤	在
	3A6x	材	罪	財	冴	坂	阪	堺	榊	肴	咲	崎	埼	碕	鷺	作	削
	3A7x	咋	搾	昨	朔	柵	窄	策	索	錯	桜	鮭	笹	匙	冊	刷	
	3B2x		察	拶	撮	擦	札	殺	薩	雑	皐	鯖	捌	錆	鮫	皿	晒
	3B3x	三	傘	参	山	惨	撒	散	桟	燦	珊	産	算	纂	蚕	讃	賛
	3B4x	酸	餐	斬	暫	残											
シ	3B4x						仕	仔	伺	使	刺	司	史	嗣	四	士	始
	3B5x	姉	姿	子	屍	市	師	志	思	指	支	孜	斯	施	旨	枝	止
	3B6x	死	氏	獅	祉	私	糸	紙	紫	肢	脂	至	視	詞	詩	試	誌
	3B7x	諮	資	賜	雌	飼	歯	事	似	侍	児	字	寺	慈	持	時	
	3C2x		次	滋	治	爾	璽	痔	磁	示	而	耳	自	蒔	辞	汐	鹿
	3C3x	式	識	鴫	竺	軸	宍	雫	七	叱	執	失	嫉	室	悉	湿	漆
	3C4x	疾	質	実	蔀	篠	偲	柴	芝	屡	蕊	縞	舎	写	射	捨	赦
	3C5x	斜	煮	社	紗	者	謝	車	遮	蛇	邪	借	勺	尺	杓	灼	爵
	3C6x	酌	釈	錫	若	寂	弱	惹	主	取	守	手	朱	殊	狩	珠	種
	3C7x	腫	趣	酒	首	儒	受	呪	寿	授	樹	綬	需	囚	収	周	
	3D2x		宗	就	州	修	愁	拾	洲	秀	秋	終	繍	習	臭	舟	蒐
	3D3x	衆	襲	讐	蹴	輯	週	酋	酬	集	醜	什	住	充	十	従	戎
	3D4x	柔	汁	渋	獣	縦	重	銃	叔	夙	宿	淑	祝	縮	粛	塾	熟
	3D5x	出	術	述	俊	峻	春	瞬	竣	舜	駿	准	循	旬	楯	殉	淳
	3D6x	準	潤	盾	純	巡	遵	醇	順	処	初	所	暑	曙	渚	庶	緒
	3D7x	署	書	薯	藷	諸	助	叙	女	序	徐	恕	鋤	除	傷	償	

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
シ	3E2x		勝	匠	升	召	哨	商	唱	嘗	奨	妾	娼	宵	将	小	少
	3E3x	尚	庄	床	廠	彰	承	抄	招	掌	捷	昇	昌	昭	晶	松	梢
	3E4x	樟	樵	沼	消	渉	湘	焼	焦	照	症	省	硝	礁	祥	称	章
	3E5x	笑	粧	紹	肖	菖	蒋	蕉	衝	裳	訟	証	詔	詳	象	賞	醤
	3E6x	鉦	鍾	鐘	障	鞘	上	丈	丞	乗	冗	剰	城	場	壌	嬢	常
	3E7x	情	擾	条	杖	浄	状	畳	穣	蒸	譲	醸	錠	嘱	埴	飾	
	3F2x		拭	植	殖	燭	織	職	色	触	食	蝕	辱	尻	伸	信	侵
	3F3x	唇	娠	寝	審	心	慎	振	新	晋	森	榛	浸	深	申	疹	真
	3F4x	神	秦	紳	臣	芯	薪	親	診	身	辛	進	針	震	人	仁	刃
	3F5x	塵	壬	尋	甚	尽	腎	訊	迅	陣	靭						
ス	3F5x											笥	諏	須	酢	図	厨
	3F6x	逗	吹	垂	帥	推	水	炊	睡	粋	翠	衰	遂	酔	錐	錘	随
	3F7x	瑞	髄	崇	嵩	数	枢	趨	雛	据	杉	椙	菅	頗	雀	裾	
	402x		澄	摺	寸												
セ	402x					世	瀬	畝	是	凄	制	勢	姓	征	性	成	政
	403x	整	星	晴	棲	栖	正	清	牲	生	盛	精	聖	声	製	西	誠
	404x	誓	請	逝	醒	青	静	斉	税	脆	隻	席	惜	戚	斥	昔	析
	405x	石	積	籍	績	脊	責	赤	跡	蹟	碩	切	拙	接	摂	折	設
	406x	窃	節	説	雪	絶	舌	蝉	仙	先	千	占	宣	専	尖	川	戦
	407x	扇	撰	栓	栴	泉	浅	洗	染	潜	煎	煽	旋	穿	箭	線	
	412x		繊	羨	腺	舛	船	薦	詮	賎	践	選	遷	銭	銑	閃	鮮
	413x	前	善	漸	然	全	禅	繕	膳	糎							
ソ	413x										噌	塑	岨	措	曾	曽	楚
	414x	狙	疏	疎	礎	祖	租	粗	素	組	蘇	訴	阻	遡	鼠	僧	創
	415x	双	叢	倉	喪	壮	奏	爽	宋	層	匝	惣	想	捜	掃	挿	掻
	416x	操	早	曹	巣	槍	槽	漕	燥	争	痩	相	窓	糟	総	綜	聡
	417x	草	荘	葬	蒼	藻	装	走	送	遭	鎗	霜	騒	像	増	憎	
	422x		臓	蔵	贈	造	促	側	則	即	息	捉	束	測	足	速	俗
	423x	属	賊	族	続	卒	袖	其	揃	存	孫	尊	損	村	遜		
タ	423x															他	多
	424x	太	汰	詑	唾	堕	妥	惰	打	柁	舵	楕	陀	駄	騨	体	堆
	425x	対	耐	岱	帯	待	怠	態	戴	替	泰	滞	胎	腿	苔	袋	貸
	426x	退	逮	隊	黛	鯛	代	台	大	第	醍	題	鷹	滝	瀧	卓	啄
	427x	宅	托	択	拓	沢	濯	琢	託	鐸	濁	諾	茸	凧	蛸	只	

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
タ	432x		叩	但	達	辰	奪	脱	巽	竪	辿	棚	谷	狸	鱈	樽	誰
	433x	丹	単	嘆	坦	担	探	旦	歎	淡	湛	炭	短	端	箪	綻	耽
	434x	胆	蛋	誕	鍛	団	壇	弾	断	暖	檀	段	男	談			
チ	434x														値	知	地
	435x	弛	恥	智	池	痴	稚	置	致	蜘	遅	馳	築	畜	竹	筑	蓄
	436x	逐	秩	窒	茶	嫡	着	中	仲	宙	忠	抽	昼	柱	注	虫	衷
	437x	註	酎	鋳	駐	樗	瀦	猪	苧	著	貯	丁	兆	凋	喋	寵	
	442x		帖	帳	庁	弔	張	彫	徴	懲	挑	暢	朝	潮	牒	町	眺
	443x	聴	脹	腸	蝶	調	諜	超	跳	銚	長	頂	鳥	勅	捗	直	朕
	444x	沈	珍	賃	鎮	陳											
ツ	444x						津	墜	椎	槌	追	鎚	痛	通	塚	栂	掴
	445x	槻	佃	漬	柘	辻	蔦	綴	鍔	椿	潰	坪	壷	嬬	紬	爪	吊
	446x	釣	鶴														
テ	446x			亭	低	停	偵	剃	貞	呈	堤	定	帝	底	庭	廷	弟
	447x	悌	抵	挺	提	梯	汀	碇	禎	程	締	艇	訂	諦	蹄	逓	
	452x		邸	鄭	釘	鼎	泥	摘	擢	敵	滴	的	笛	適	鏑	溺	哲
	453x	徹	撤	轍	迭	鉄	典	填	天	展	店	添	纏	甜	貼	転	顛
	454x	点	伝	殿	澱	田	電										
ト	454x							兎	吐	堵	塗	妬	屠	徒	斗	杜	渡
	455x	登	菟	賭	途	都	鍍	砥	砺	努	度	土	奴	怒	倒	党	冬
	456x	凍	刀	唐	塔	塘	套	宕	島	嶋	悼	投	搭	東	桃	梼	棟
	457x	盗	淘	湯	涛	灯	燈	当	痘	祷	等	答	筒	糖	統	到	
	462x		董	蕩	藤	討	謄	豆	踏	逃	透	鐙	陶	頭	騰	闘	働
	463x	動	同	堂	導	憧	撞	洞	瞳	童	胴	萄	道	銅	峠	鴇	匿
	464x	得	徳	涜	特	督	禿	篤	毒	独	読	栃	橡	凸	突	椴	届
	465x	鳶	苫	寅	酉	瀞	噸	屯	惇	敦	沌	豚	遁	頓	呑	曇	鈍
ナ	466x	奈	那	内	乍	凪	薙	謎	灘	捺	鍋	楢	馴	縄	畷	南	楠
	467x	軟	難	汝													
ニ	467x				二	尼	弐	邇	匂	賑	肉	虹	廿	日	乳	入	
	472x		如	尿	韮	任	妊	忍	認								
ヌ	472x									濡							

		０	１	２	３	４	５	６	７	８	９	Ａ	Ｂ	Ｃ	Ｄ	Ｅ	Ｆ
ネ	４７２x										禰	祢	寧	葱	猫	熱	年
	４７３x	念	捻	撚	燃	粘											
ノ	４７３x						乃	廼	之	埜	嚢	悩	濃	納	能	脳	膿
	４７４x	農	覗	蚤													
ハ	４７４x				巴	把	播	覇	杷	波	派	琶	破	婆	罵	芭	馬
	４７５x	俳	廃	拝	排	敗	杯	盃	牌	背	肺	輩	配	倍	培	媒	梅
	４７６x	楳	煤	狽	買	売	賠	陪	這	蝿	秤	矧	萩	伯	剥	博	拍
	４７７x	柏	泊	白	箔	粕	舶	薄	迫	曝	漠	爆	縛	莫	駁	麦	
	４８２x		函	箱	硲	箸	肇	筈	櫨	幡	肌	畑	畠	八	鉢	溌	発
	４８３x	醗	髪	伐	罰	抜	筏	閥	鳩	噺	塙	蛤	隼	伴	判	半	反
	４８４x	叛	帆	搬	斑	板	氾	汎	版	犯	班	畔	繁	般	藩	販	範
	４８５x	釆	煩	頒	飯	挽	晩	番	盤	磐	蕃	蛮					
ヒ	４８５x												匪	卑	否	妃	庇
	４８６x	彼	悲	扉	批	披	斐	比	泌	疲	皮	碑	秘	緋	罷	肥	被
	４８７x	誹	費	避	非	飛	樋	簸	備	尾	微	枇	毘	琵	眉	美	
	４９２x		鼻	柊	稗	匹	疋	髭	彦	膝	菱	肘	弼	必	畢	筆	逼
	４９３x	檜	姫	媛	紐	百	謬	俵	彪	標	氷	漂	瓢	票	表	評	豹
	４９４x	廟	描	病	秒	苗	錨	鋲	蒜	蛭	鰭	品	彬	斌	浜	瀕	貧
	４９５x	賓	頻	敏	瓶												
フ	４９５x					不	付	埠	夫	婦	富	冨	布	府	怖	扶	敷
	４９６x	斧	普	浮	父	符	腐	膚	芙	譜	負	賦	赴	阜	附	侮	撫
	４９７x	武	舞	葡	蕪	部	封	楓	風	葺	蕗	伏	副	復	幅	服	
	４Ａ２x		福	腹	複	覆	淵	弗	払	沸	仏	物	鮒	分	吻	噴	墳
	４Ａ３x	憤	扮	焚	奮	粉	糞	粉	雰	文	聞						
ヘ	４Ａ３x											丙	併	兵	塀	幣	平
	４Ａ４x	弊	柄	並	蔽	閉	陛	米	頁	僻	壁	癖	碧	別	瞥	蔑	箆
	４Ａ５x	偏	変	片	篇	編	辺	返	遍	便	勉	娩	弁	鞭			

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ホ	4A5x														保	舗	鋪
	4A6x	圃	捕	歩	甫	補	輔	穂	募	墓	慕	戊	暮	母	簿	菩	倣
	4A7x	俸	包	呆	報	奉	宝	峰	峯	崩	庖	抱	捧	放	方	朋	
	4B2x		法	泡	烹	砲	縫	胞	芳	萌	蓬	蜂	褒	訪	豊	邦	鋒
	4B3x	飽	鳳	鵬	乏	亡	傍	剖	坊	妨	帽	忘	忙	房	暴	望	某
	4B4x	棒	冒	紡	肪	膨	謀	貌	貿	鉾	防	吠	頬	北	僕	卜	墨
	4B5x	撲	朴	牧	睦	穆	釦	勃	没	殆	堀	幌	奔	本	翻	凡	盆
マ	4B6x	摩	磨	魔	麻	埋	妹	昧	枚	毎	哩	槙	幕	膜	枕	鮪	柾
	4B7x	鱒	桝	亦	俣	又	抹	末	沫	迄	侭	繭	麿	万	慢	満	
	4C2x		漫	蔓													
ミ	4C2x				味	未	魅	巳	箕	岬	密	蜜	湊	蓑	稔	脈	妙
	4C3x	粍	民	眠													
ム	4C3x				務	夢	無	牟	矛	霧	鵡	椋	婿	娘			
メ	4C3x														冥	名	命
	4C4x	明	盟	迷	銘	鳴	姪	牝	滅	免	棉	綿	緬	面	麺		
モ	4C4x															摸	模
	4C5x	茂	妄	孟	毛	猛	盲	網	耗	蒙	儲	木	黙	目	杢	勿	餅
	4C6x	尤	戻	籾	貰	問	悶	紋	門	匁							
ヤ	4C6x										也	冶	夜	爺	耶	野	弥
	4C7x	矢	厄	役	約	薬	訳	躍	靖	柳	薮	鑓					
ユ	4C7x												愉	愈	油	癒	
	4D2x		諭	輸	唯	佑	優	勇	友	宥	幽	悠	憂	揖	有	柚	湧
	4D3x	涌	猶	猷	由	祐	裕	誘	遊	邑	郵	雄	融	夕			
ヨ	4D3x														予	余	与
	4D4x	誉	輿	預	傭	幼	妖	容	庸	揚	揺	擁	曜	楊	様	洋	溶
	4D5x	熔	用	窯	羊	耀	葉	蓉	要	謡	踊	遥	陽	養	慾	抑	欲
	4D6x	沃	浴	翌	翼	淀											

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ラ	4D6x						羅	螺	裸	来	萊	頼	雷	洛	絡	落	酪
	4D7x	乱	卵	嵐	欄	濫	藍	蘭	覧								
リ	4D7x									利	吏	履	李	梨	理	璃	
	4E2x		痢	裏	裡	里	離	陸	律	率	立	葎	掠	略	劉	流	溜
	4E3x	琉	留	硫	粒	隆	竜	龍	侶	慮	旅	虜	了	亮	僚	両	凌
	4E4x	寮	料	梁	涼	猟	療	瞭	稜	糧	良	諒	遼	量	陵	領	力
	4E5x	緑	倫	厘	林	淋	燐	琳	臨	輪	隣	鱗	麟				
ル	4E5x													瑠	塁	涙	累
	4E6x	類															
レ	4E6x		令	伶	例	冷	励	嶺	怜	玲	礼	苓	鈴	隷	零	霊	麗
	4E7x	齢	暦	歴	列	劣	烈	裂	廉	恋	憐	漣	煉	簾	練	聯	
	4F2x		蓮	連	錬												
ロ	4F2x					呂	魯	櫓	炉	賂	路	露	労	婁	廊	弄	朗
	4F3x	楼	榔	浪	漏	牢	狼	篭	老	聾	蝋	郎	六	麓	禄	肋	録
	4F4x	論															
ワ	4F4x		倭	和	話	歪	賄	脇	惑	枠	鷲	亙	亘	鰐	詫	藁	蕨
	4F5x	椀	湾	碗	腕												
	4F5x																

参　考　　**漢字パターンのドットとデータの対応**

漢字パターン（16×16ドット）を32個のデータで表わしていますデータは、次の順に入力されます。

順番	データ 7	6	5	4	3	2	1	0	データ 7	6	5	4	3	2	1	0	順番
1								■				■					9
2		■			■	■	■	■	■	■	■	■	■	■	■		10
3			■					■				■					11
4																	12
5	■					■	■	■	■	■	■	■	■	■			13
6		■				■				■				■			14
7						■				■				■			15
8						■	■	■	■	■	■	■	■	■			16
17				■						■							25
18				■		■	■	■	■	■	■	■	■	■			26
19			■							■							27
20			■							■							28
21		■			■	■	■	■	■	■	■	■	■	■	■		29
22		■						■				■					30
23	■						■						■				31
24					■	■								■	■		32